201-395
203-165
A

Attorney's Docket No. **831-2**

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Assistant Commissioner for Patents
Box Patent Application
Washington, D.C. 20231

UTILITY PATENT APPLICATION TRANSMITTAL

Sir:

Transmitted herewith for filing is the patent application of:

First Named Applicant (or Application Identifier): **Luciano F. Paone**

Title of Application: **COMPUTER IMPLEMENTED SECRET OBJECT KEY BLOCK CIPHER ENCRYPTION AND DIGITAL SIGNATURE DEVICE AND METHOD**

1. Type of Application (37 C.F.R. 1.53(b))

This application is a(n):

[x] Original (nonprovisional) application.

[ ] Continuing application:

[ ] Divisional [ ] Continuation [ ] Continuation-in-Part (CIP)

of Serial No. 08/____, filed on ________.

CERTIFICATION UNDER 37 CFR 1.10

I hereby certify that this New Application Transmittal and the documents referred to as enclosed herein are being deposited with the United States Postal Service on this date, **December 12, 1997**, in an envelope as "Express Mail to Addressee" Mailing Label Number **EM331055149US**, addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Susan L. Toledano
Name of person mailing paper

Susan L. Toledano
Signature of person mailing paper

2. **Enclosed Papers Required to Obtain Application Filing Date under 37 CFR 1.53(b)**

<u>23</u> Pages of specification
<u>12</u> Pages of claims
<u>1</u> Pages of Abstract
<u>25</u> Sheets of drawings [ ] Formal [x] Informal

3. **Oath or Declaration**

[x] An Oath or Declaration is enclosed.

[ ] A copy of the Declaration or Oath filed in prior application **0 /** ______, is enclosed.

[ ] The entire disclosure of the prior application, from which a copy of the oath or Declaration is supplied, is considered as being a part of the disclosure of the accompanying application and is hereby incorporated by reference.

4. **Additional Papers Enclosed**

[X] Return Receipt Postcard (specifically itemized) (M.P.E.P. § 503)

[ ] Preliminary Amendment.

[ ] Information Disclosure Statement (37 CFR 1.98).

[ ] Form PTO-1449 [ ] Copies of IDS Citations

[ ] Nucleotide and/or Amino Acid Sequence Listing computer-readable copy, paper copy, and statement verifying identity of computer-readable and paper copies.

[ ] Certified Copy of Priority Document(s)

[ ] Verified translation of non-English language application (37 C.F.R. 1.52(d).

[ ] Other: ______________________________

5. **Assignment**

An assignment of the invention to <u>Paonet Software, Inc.</u> ______________________________ :

[x] is enclosed, with an Assignment Recordation Cover Sheet (Form PTO-1595).

[ ] was made in prior application No. **08/** ______, filed on ____________.

[ ] A copy of the Assignment (and any recordation cover sheet) is enclosed.

6. Fee Calculation (37 CFR 1.16)

Regular Application (37 CFR 1.16(a)) Basic Fee $790.00

FEES FOR CLAIMS AS FILED											
Number filed				Number extra		Rate				
Total Claims (37 CFR 1.16(c))	36	-	20	=	16	X	$ 22.00	=	$	352.00
Independent Claims (37 CFR 1.16(b))	3	-	3	=	0	X	$ 82.00	=	$	0
Multiple Dependent Claims (37 CFR 1.16(d))						+	$270.00	=	$	

Fee Calculation for Extra Claims $ **352.00**

[ ] Amendment canceling extra claims enclosed.

[ ] Amendment deleting multiple-dependencies enclosed.

Total Filing Fee Calculation $ **1,142.00**

7. Small Entity Statement

[x] A Verified Statement that this is a filing by a small entity under 37 CFR 1.9 and 1.27;

[x] is enclosed. [ ] will follow.

[ ] Status as a small entity was claimed in prior application **08/**____, from which benefit is being claimed for this application under:

- [ ] 35 U.S.C. 119(e),
- [ ] 35 U.S.C. 120,
- [ ] 35 U.S.C. 121,
- [ ] 35 U.S.C. 365(c),

and which status as a small entity is still proper and desired.

[ ] A copy of the verified statement in the prior application is enclosed.

Filing Fee Calculation (50% of Filing Fee calculated in Item 6 above) $ **571.00**

8. Fee Payment

[ ] Not enclosed. No filing fee is to be paid at this time.

[x] Enclosed:

[x]	Basic filing fee (Item 6 or 7 above)	$ **571.00**
[ ]	Fee for recording Assignment ($40.00 (37 CFR 1.21(h)))	$ **40.00**
[ ]	Processing and retention fee ($130.00 (37 CFR 1.53(d) and 1.21(l)))	$________
	Total fees enclosed	$ **611.00**.

9. Method of Payment of Fees

[x] Check in the amount of $ **611.00**________.

[ ] Charge Deposit Account No. 08-2461 in the amount of $____________.
A duplicate of this transmittal is enclosed.

10. Authorization to Charge Additional Fees

[X] The Commissioner is hereby authorized to charge the following additional fees by this paper and during the entire pendency of this application to Deposit Account No. 08-2461:

[X] 37 CFR 1.16(a), (f), or (g) (filing fees)

[X] 37 CFR 1.16(b), (c), and (d) (presentation of extra claims)

[X] 37 CFR 1.16(e) (surcharge for filing the basic fee and/or declaration at a date later than the filing date of the application)

[X] 37 CFR 1.17 (application processing fees)

A duplicate of this transmittal is enclosed.

11. Instructions as to Overpayment

[X] Credit Deposit Account 08-2461. [ ] Refund.

12. Correspondence Address

Please address all correspondence to:

Glenn T. Henneberger, Esq.
HOFFMANN & BARON, LLP
350 Jericho Turnpike
Jericho, New York 11753

Telephone: (516) 822-3550 Fax: (516) 822-3582

Glenn T. Henneberger
Registration No. 36,074
Attorney for Applicant(s)

GTH/slt

COMPUTER IMPLEMENTED SECRET OBJECT KEY BLOCK CIPHER ENCRYPTION AND DIGITAL SIGNATURE DEVICE AND METHOD

Inventor: Luciano F. Paone
18 Knobhill Drive
Kings Park, New York 11754

COMPUTER IMPLEMENTED SECRET OBJECT KEY BLOCK CIPHER ENCRYPTION AND DIGITAL SIGNATURE DEVICE AND METHOD

BACKGROUND OF THE INVENTION

Field Of The Invention

This invention relates generally to a computer implemented device and method for cryptography and, more particularly relates to a computer implemented block cipher encryption method utilizing dynamic device keys and a digital signature.

DESCRIPTION OF THE PRIOR ART

The principal goal of encryption is to render communicated data secure from unauthorized eavesdropping. This is generally referred to as the "secrecy" or "confidentiality" requirement of cryptographic systems. A related requirement is the "authenticity" or "integrity" requirement, which ensures that the communicated information is authentic, i.e. that it has not been tampered with, either deliberately or inadvertently. For purposes of further discussion, some definitions are provided.

“Plaintext” is used to refer to a message before encrypting and after decrypting by a cryptographic system. “Ciphertext” is the form that the encrypted part of the message takes during transmission over a communications channel or within a computer memory storage device. “Encryption” is the process of transformation from plaintext to ciphertext. “Decryption” is the process of transformation from ciphertext to plaintext. Both encryption and decryption are controlled by keys. Without knowledge of the encryption key, a message cannot be encrypted, even with knowledge of the encrypting process. Similarly, without knowledge of the decryption key, the message cannot be decrypted, even with knowledge of the decrypting process.

Data encryption processes scramble plaintext data into ciphertext to prevent unauthorized access to the data. Decryption processes restore the plaintext from the ciphertext (encrypted data). Symmetric key encryption processes utilize the same key for encryption and decryption.

In order to ensure the integrity of the ciphertext, the encryption must be sufficiently complex to prevent unauthorized access to the encrypted data. Two current methods of cryptanalytic attacks, methods designed to break encryption processes, are linear and differential cryptanalysis. In linear cryptanalysis, linear approximations of the block cipher and key schedule used in the encryption process are constructed. Collected plaintexts and associated ciphertexts are used to exploit a bias in the encryption process and key schedule. If the bias is very small many plaintexts and associated ciphertext pairs are used. This method tries different keys and eventually converges on the correct key. In differential

cryptanalysis, pairs of ciphertexts whose plaintexts have particular differences are compared. The evolution of these differences as the plaintexts propagate through the rounds of the encryption process when they are encrypted with the same key are analyzed. Different probabilities are assigned to different keys. As more and more ciphertext pairs are analyzed, the method converges on the correct key. Both cryptanalytic attacks exploit the fact that existing symmetric key block cipher encryption processes use static keys to create the ciphertext.

Still other cryptanalytic attacks exploit the fact that current encryption processes use small key spaces and small block sizes. For certain key spaces and block sizes, current data storage technologies now make it possible to store all combinations of an encrypted block with an associated key for a chosen plaintext block. Thus, breaking the encryption process merely involves a quick look-up table. Accordingly, the present invention seeks to overcome the disadvantages associated with currently available encryption methods and create ciphertext which is very secure and substantially immune to known cryptanalytic attacks.

OBJECTS AND SUMMARY OF THE INVENTION

It is an object of the present invention to provide a computer implemented encryption device and method which is substantially immune to currently available cryptanalytic attacks using an object key comprised of data and methods which operate on the data.

It is another object of the present invention to provide a computer implemented block cipher encryption device and method using an object key which is dynamic, i.e., changing throughout the encryption process.

It is still a further object of the present invention to provide a computer implemented block cipher encryption device and method using a dynamic object key which is modified by a random session object key.

It is yet another object of the present invention to provide a computer implemented block cipher encryption device and method such that each input plaintext data block is encrypted using a new key schedule to create the ciphertext.

It is a further object of the present invention to provide a computer implemented encryption device and method which includes a digital signature appended to the ciphertext, the digital signature being unique to the data being signed.

It is yet another object of the present invention to provide a computer implemented encryption device and method using a secret key composed of two 2048-bit user object keys and 512-bit random session object key.

The present invention provides a computer implemented data encryption device and method utilizing object keys (consisting of data and methods that operate on the data), a large key space and a large block size. The object keys (K_OBJECT) are dynamic keys and are

composed of a 4096-bit static initial state that is created by the user and a method that modifies the keys based on seeding from a random session object key (R_OBJECT). The key modification is performed for each input plaintext data block so that each data block is encrypted with a different key. The initial state of the object key is used in the block cipher encryption process to encrypt a 512-bit random session key. The random session object key is used as the initial state of the random session object key (R_OBJECT). The running states of the random session object key (R_OBJECT) seed the object key (K_OBJECT) modification methods. The object key's (K_OBJECT) running state provides an index for seeding of the random session object key's (R_OBJECT) modification method. The encryption process utilizes a 512-bit (64 byte) input block size.

The object key (K_OBJECT) is preferably composed of two 2048-bit sub-object keys (K1 and K2). The two sub-object keys are used as inputs to an expansion function that provides 13584 bytes for the block's key schedule. Similar to the object key (K_OBJECT), the block's key schedule is regenerated anew for each plaintext input block.

The encryption process includes many linear and nonlinear-keyed operations. The keyed operations include addition, 32-bit sliding window rotation, bit-wise exclusive or, 8-bit by 8-bit substitution using different transverse counts for each substitution, byte transposition and bit transposition. All operations are nested and repeated multiple times.

The decryption process is identical to the encryption process with the exception that the keyed encryption operations on the data are run in reverse.

In order to authenticate an encrypted file, a digital signature is provided. To create the digital signature, the ciphertext is used as input into a 2048-bit object keyed one-way hash function to produce a 2048-bit digital signature that is appended to the ciphertext. The 2048-bit digital signature object key is seeded with the output of hash rounds. A third party or secured server can execute verification software that contains the user's secret digital signature key to regenerate the digital signature of the ciphertext that was signed and compare it to the signature that is appended to the ciphertext to determine if they match. The plaintext is not compromised and only the party that contains the receiver's encryption key can create ciphertext that the receiver can decrypt into plaintext.

A preferred form of the encryption device and method, as well as other embodiments, objects, features and advantages of this invention will be apparent from the following detailed description of illustrative embodiments thereof, which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a simplified block diagram of a computer based system capable of implementing the encryption method of the present invention.

Figure 2 is a simplified flow chart representation of the installation and initialization of the encryption software onto a computer based system of Figure 1.

Figure 3 is a simplified flow chart representation of the encryption process performed by the computer system formed in accordance with the present invention.

Figure 4 is a simplified flow chart representation of creating the digital signature to be appended to ciphertext created by the encryption process of Figure 3.

Figure 5 is a flow chart representation of the steps carried out by the computer system to create ciphertext.

Figure 6 is a flow chart representation of the switch key function formed in accordance with the present invention.

Figure 7 is a flow chart representation of the modification function of the sub-object key K1 based on seeding from the random session object key K3.

Figure 8 is a flow chart representation of the modification function of the sub-object key K2 based on seeding from the random session object key K3.

Figure 9 is a flow chart representation of the methods to create unique key schedules for each block of plaintext to be encrypted in accordance with the present invention.

Figure 10 is a flow chart representation of the method to create a digital signature for each encrypted file formed in accordance with the present invention.

Figure 11 is a flow chart representation of the modification method of the object key used in generating a digital signature formed in accordance with the present invention.

Figure 12 is a flow chart representation of the method of producing a digital signature in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring initially to Figure 1, a simplified block diagram of a computer based system 10 used for implementing the encryption method of the present invention is illustrated. In general, the encryption method of the present invention may be implemented in software and embodied on a magnetic storage device 2 such as a computer diskette. The diskette may be inserted and into and read by a disk drive 4 for execution by the computer system 10. The computer system 10 generally includes a display device 6, such as a CRT display, a keyboard 8 for entering information, a pointing device 12, and a printer 14. Internally, the computer system 10 includes a central processing unit (CPU) 16 which preferably includes an internal clock function, a memory device, such as a random access memory (RAM) and an output port which may be connected to a modem for remote access to other computer systems within a network. The various computer system components are operatively coupled and communicate via a system bus 24, or similar hardware architecture.

It will be appreciated by those of ordinary skill in the art that the computer system illustrated in Figure 1 is merely representative of one form of computer based system capable of carrying out the functions of the encryption method of the present invention. For example,

the encryption method may be executed on a single computer workstation to protect information stored therein or in a cryptographic communications system including a plurality of computer stations cooperatively coupled and operating via a computer network to protect information being transmitted and received via the network.

Further, it is to be understood that the individual system components may vary in type, while still providing similar functions. For example, the pointing device 12, used to position a display cursor and select certain functions and/or items displayed on the display device may be in the form of a mouse, trackball or touchscreen. In a preferred embodiment, the user is provided with a keyboard 8 for data entry and key initialization and a mouse as the pointing device 12.

Regardless of the computer system being used, the encryption method of the present invention may be loaded from the diskette into the disk drive 4 and executed by the CPU 16 in conjunction with the RAM 18. In this manner, the functions of the encryption method can be executed to create ciphertext (encrypted data) from input plaintext, or to decrypt ciphertext to restore the input plaintext.

Referring now to Figure 2, the installation and initialization of the encryption method is illustrated in flow chart format. The encryption software is initially installed 26 into the disk drive 4 and executed by the CPU 16 and RAM 18. Once installed, the user types a series of random key strokes 28. Based upon the random keystrokes, an initial state of the object key (K_OBJECT comprising sub-object key K1 and sub-object key K2) is created 30. At this

point, the user creates a password 32 which becomes associated or linked with the initial state of the object key. The initial state of the object key is appended with a checksum and encrypted 34 along with the user's password. The user then creates a password for a remote user 36 to allow decryption of transmitted ciphertext. The remote user password is also appended with a checksum and encrypted 38. With this initialization procedure completed, the user is ready to create ciphertext from plaintext data using the encryption method of the present invention.

The computer implemented encryption device and method of the present invention utilizes object keys, consisting of data and methods that operate on the data, a large key space and a large block size. More specifically, the present invention discloses a 4096-bit secret object key block cipher encryption method utilizing a 2048-bit object keyed digital signature process to sign encrypted files. In the object key, the data comprises a set of binary digits that can represent two to the power of the bit length possible combinations. The encryption method uses a 512-bit block and the object key is a dynamic key. The object key (K_OBJECT) contains key modification methods that mutate the key's state based on seeding from a random session object key (R_OBJECT). The random session object key also consists of data and methods that operate on the data. As illustrated in Figure 1, the user of the encryption method creates the object key's 4096-bit initial state.

In order to create an encrypted file, the input plaintext file is compressed using a redundant byte reducing method and padded with random bytes to produce a file with a length being a multiple of 512-bits (64 bytes) 40. Using a time clock which is a part of the

computer system CPU, a 512-bit random number is generated and assigned as an initial state of the random session object key (R_OBJECT) 42.

Next, a switch key is created 44, from an initial state of the object key (K_OBJECT comprising K1 and K2). The function of the switch key will be discussed later in further detail. In the block cipher encryption method in accordance with the present invention, the plaintext blocks are encrypted using a dynamic key schedule. The encryption key schedule is created 46 from an initial state of the object key (K_OBJECT comprising K1 and K2). As will become apparent, the encryption method of the present invention encrypts each plaintext data block utilizing a different key schedule to produce ciphertext that is immune to current cryptanalytic attacks designed to work with static key schedules.

Using the initial state of the key schedule, the initial state of the random session object key is encrypted 48. At this point in the encryption method of the present invention, the random session object key is modified 50 based on seeding from the object key (K_OBJECT using K2 only). This modification of the random session object key is the first step within a logic loop to determine when the entire plaintext file has been transformed into ciphertext. In the next step, the state of the object key (K_OBJECT comprising K1 and K2) is modified based on seeding from the random session object key 52. Using the modified object key from the previous step, a new key schedule is created 54. The new key schedule is utilized to encrypt the input plaintext data block using the modified object key (K_OBJECT comprising K1 and K2) 56. At this point, the encryption method includes a decision box to determine if further plaintext data blocks exist to be encrypted 58. If further plaintext data blocks exist,

the method returns to the step in which the random session object key is modified based on seeding from (K_OBJECT using K2 only) 50. Accordingly, new states for the random session object key (step 50), the object key (K_OBJECT comprising K1 and K2) (step 52), and a new key schedule (step 54) are created for each plaintext data block which is encrypted. Once all plaintext data blocks are encrypted, the encryption method of the present invention transposes 60 the encrypted data using the switch key created in step 44. More specifically, a final 128 byte transposition is performed on the encrypted data with the first 128 bytes of ciphertext transpositioned within the entire ciphertext file with respect to the switch key. After transposition, the encryption is completed 62.

In the preferred embodiment of the present invention, a digital signature is produced and appended to the ciphertext. A digital signature allows a third party or secured server to execute verification software that contains the original user's secret digital signature key to regenerate the digital signature of the ciphertext and compare it to the signature that was appended to the encrypted ciphertext to determine if they match. In this manner, the plaintext is not compromised and only the party that has access to the receiver's encryption key can create ciphertext that the receiver can decrypt into plaintext.

The unique method for creating the digital signature is illustrated in the flow chart of Figure 4. The encrypted data file or ciphertext is input into a 2048-bit object keyed one-way hash function 64. A 2048-bit digital signature is produced from the ciphertext along with a 2048-bit object key specific to the particular input file 60. The 2048-bit digital signature is

appended to the encrypted data or ciphertext. The 2048-bit digital signature object key is seeded with the output of hash rounds to create the digital signatures for each plaintext file.

As previously noted, the computer implemented encryption device and method of the present invention preferably uses an object key (K_OBJECT) which comprises two 2048-bit sub-object keys (K1; K2). The two sub-object keys are used as inputs to an expansion function that provides 13,584 bytes for the cipher block's key schedule (KS).

More specifically, Figure 5 is a flow chart representation of the process to create ciphertext using an object key comprising two sub-object keys and a random session object key. In Figure 5, the object key is made up of two 2048-bit sub-object keys K1 and K2, respectively. The random session object key is represented by K3 (K3=srand((KS[i]xKS[i]+KS[1])^ time (NULL); K3[i] =((rand ()%255)+1+High_resolution_timer) where "i" is the running index) and KS is the key schedule used to encrypt each plaintext input block of 64 bytes. As earlier described, the object keys K1 and K2 are composed of initial states that are created by the user and functions that modify the keys for each plaintext input block.

Referring to Figure 5, the input file is compressed using a redundant byte reducing method and padded 72 with random bytes to produce a file with a length having a multiple of 512-bits (64 bytes). The encryption process extensively uses linear and non-linear keyed operations on the plaintext to produce the ciphertext. A substitution array 74 consisting of 256 unique 8-bit elements and a transverse array 76 consisting of 64 unique 8-bit elements

are continuously transpositioned with respect to the current key schedule. The two arrays 74, 76 provide an 8-bit by 8-bit S-box 78 (substitution process) that contains a transverse count substitution process for each input into the S-box. More specifically, the steps 74, 76 and 78 provide a transposition of a sequence of integers 0-63 with respect to a key and provides a count of substitution rounds for a particular input entering the S-box. Continuing the method, a nested keyed addition 80, sliding window rotation 82, bit-wise exclusive or 84, 8-bit by 8-bit transverse repeated substitution 86, 88, byte transposition 90 and bit transposition encryption process 92 are executed by the computer system. The sliding window rotation operates on a 32-bit boundary incrementing (sliding) one byte (8-bits) after each rotation are used to create the ciphertext. The outer loop cycles four times with the inner loop cycling four times for each outer loop iteration. After each block has been processed, the key modification methods in the object key (K_OBJECT) and the random session object key (R_OBJECT) are performed to modify the object key's and the random session object key's state. A new key schedule is created after the modification methods are performed. The key schedule creation process is illustrated in Figure 9. In Figure 9, the two 2048-bit object key states are used to create a 13,584 byte key schedule for the encryption process.

Referring to Figure 9, the key schedule is created with multiple linear congruential generators. Three offset variables are utilized. Offset_one is initialized to 0, offset_two is initialized to 1 and offset_three is initialized to 2. The first byte of key schedule is set to K1[K2[#]] + K2[K1[#]] (Step 136). In Figure 9, the letter "a" is a multiplier, "b" is an offset and "a_prev" is the previous value of "a". In step 138, a_prev is initialized. The first linear congruential generator uses K1[index] as a multiplier and K2[index + offset_one] as an offset

(Step 140). The index is a running counter. The second linear congruential generator uses K2[index + offset_two] as a multiplier and K1[index + offset_three as an offset (step 142). The third linear congruential generator produces the key schedule. The third linear congruential generator uses the output of the first linear congruential generator as a multiplier and the output of the second linear congruential generator as an offset (step 144). Offset_one, offset_two and offset_three are incremented every 256 rounds (steps 146, 148). In the preferred embodiment, the linear congruential generators are run through 13,584 rounds to produce a 13,584 byte key schedule. (KS as shown in Figure 5).

After all the plaintext blocks have been processed a final 128-byte transposition 98 is performed with the first 128 bytes of ciphertext transpositioned within the entire ciphertext file with respect to a switch-key (SWK). As illustrated in Figure 6, the switch-key is created from the initial state of the object key. The switch key is initialized with elements of the initial state of the object key. The switch key is grouped by 32-bit blocks and the current switch key element is replaced using the following steps:

the current switch key element is bit-wise exclusive ored to a switch key element indexed two elements from the current element (step 101);

the output of the previous operation is rotated to the right switch key indexed three elements from the current element modulus thirty-one plus one (step 103);

the output of the previous operation is bit-wise exclusive ored to a switch key element indexed three elements from the current element (step 105); and

the previous three steps are repeated for each final transposition switch operation. In an alternative embodiment, the steps 101, 103, 105 may include a hashing function in the creation of the switch key.

Referring back to Figure 5, the plaintext file extension and a checksum are appended to the ciphertext. Delimiters are used to mark the beginning and end of the ciphertext. This completes the encryption process to create the ciphertext, which, in a preferred embodiment, will include a digital signature appended thereto.

As earlier noted, the object key is dynamic and changes with each block of input data. The first sub-object key's modification method, function F1 of Figure 5, is illustrated in Figure 7. In the first execution of the modification method, the random session key's elements are used instead of the running index "s". The first sub-object key (K1) has a modification method which includes the following iterations:

performing a bit-wise exclusive or on an unsigned byte of the random session object key(K3) to an unsigned byte of the current state of the object key provided by an incremented index into the current state of the object key (I_BYTE_OBJECT_KEY) (step 104);

performing an unsigned byte addition on the output byte of the previous operation (PREV_OUTPUT) with I_BYTE_OBJECT_KEY (step 106);

performing a 16-bit multiplication of PREV_OUTPUT and I_BYTE_OBJECT_KEY modulus 254 and add 2 (step 108);

performing a 16-bit addition of PREV_OUTPUT and I_BYTE_OBJECT_KEY (step 110);

performing another 16-bit addition of PREV_OUTPUT and I_BYTE_OBJECT_KEY (step 112);

performing a bit-wise exclusive or of PREV_OUTPUT with a 16-bit unsigned integer of the current state of the object key provided by an incremented index into the current state of the object key (I_INT_OBJECT_KEY) (step 114);

rotating PREV_OUTPUT to the right I_BYTE_OBJECT_KEY modulus 15 plus 1 times (step 116);

performing a 16-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY (step 118);

performing a 16-bit multiplication of PREV_OUTPUT and I_INT_OBJECT_KEY with the lower order byte of I_INT_OBJECT_KEY modulus 254 plus 2 (step 120);

performing a 16-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY (step 122);

performing another 16-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY (step 124);

performing a bit-wise exclusive or of PREV_OUTPUT with a 32-bit unsigned long integer of the current state of the object key provided by an incremented index into the current state of the object key (I_LONG_INT_OBJECT_KEY) (step 126);

rotating PREV_OUTPUT to the left I_BYTE_OBJECT_KEY modulus 31 plus 1 times (step 128);

performing a bit-wise exclusive or of PREV_OUTPUT with I_LONG_INT_OBJECT_KEY (step 130);

repeating the previous set of operations eighty-four times substituting the random seed unsigned byte with a byte from the four byte output block provided by the previous set of operations recursively setting the current output block to the next output block when the current output block is exhausted and utilizing a different ordered byte each round (step 132);

performing a byte transposition of the bytes in the new 256 byte output block (N_OUTPUT) provided by the previous set of operations utilizing the following operation:

performing a byte-wise index through N_OUTPUT and switching the current byte of N_OUTPUT with the N_OUTPUT byte indexed at position I_BYTE_OBJECT_KEY, indexing through the entire block of N_OUTPUT (step 134). This modification method makes the ciphertext generated using the dynamic object key immune from cryptanalytic attacks.

The second sub-object key's (K2) modification method is illustrated in Figure 8. The second sub-object key's modification method is similar to the first sub-object key's modification method with the exception that the rotations are performed in the reverse. Accordingly, further discussions of the flow chart illustrated in Figure 8 is not necessary.

Referring now to Figure 10, the eight hash functions used to create the 2048-bit object keyed digital signature is illustrated 150. In the preferred embodiment, a digital signature is created and appended to each input plaintext file transformed into ciphertext for authentication purposes. The hash functions illustrated in Figure 10 are one-way hash

functions. Referring to Figure 12, the process for generating the digital signatures for a file under control of an object key using keyed hashing of the input ciphertext blocks includes the following iterations:

dividing the input data into 256 byte data blocks (step 200);

further dividing the data block into 64 32-bit blocks (VAR_BLOCK) (step 202);

modifying each VAR_BLOCK and element of the key by a plurality of unique one-way irreversible hash function (step 202); (It should be noted that before each VAR_BLOCK modification, the object key used to create the digital signature is modified as illustrated in Figure 11 (Steps 205).

repeating the previous steps for all VAR_BLOCK hash function combinations to create an output running message digest (steps 204);

performing a bit-wise exclusive or of the running message digest to the next input data block (step 206);

repeating the previous four steps for all data blocks (step 208);

transpositioning each byte of an output from the previous step by switching a position of each byte with another byte at a position provided by an element of the key wherein the position provided by an element of the key is bounded by the size of the data block (step 210);

appending a checksum to the digital signature (step 212); and

appending the digital signature to the ciphertext (step 214).

Delimiters are used to mark the beginning and end of the digital signature. The digital signature generated is appended to the ciphertext to provide for verification by a third party or secured server as earlier described.

Figure 11 illustrates the digital signature's object key modification method. The digital signature's object key modification method is composed of the following:

performing an 8-bit addition or of a seed to an unsigned byte of a current state of the object key provided by an incremented index into the current state of the object key (I_BYTE_OBJECT_KEY) (step 159);

performing an unsigned byte addition on the output byte of the previous operation (PREV_OUTPUT) with I_BYTE_OBJECT_KEY (step 160);

performing a bit-wise exclusive or of PREV_OUTPUT to I_BYTE_OBJECT_KEY (step 162);

performing a 16-bit multiplication of PREV_OUTPUT and I_BYTE_OBJECT_KEY modulus 254 and add 2 (step 164);

performing a 16-bit addition of PREV_OUTPUT and I_BYTE_OBJECT_KEY (step 166);

performing another 16-bit addition of PREV_OUTPUT and I_BYTE_OBJECT_KEY (step 168);

performing another 16-bit addition of PREV_OUTPUT and I_BYTE_OBJECT_KEY (step 170);

rotating PREV_OUTPUT to the left I_BYTE_OBJECT_KEY modulus 15 plus 1 times (step 172);

performing a 16-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY (step 174);

performing a 16-bit multiplication of PREV_OUTPUT and I_INT_OBJECT_KEY with the lower order byte of I_INT_OBJECT_KEY modulus 254 plus 2 (step 176);

performing a 16-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY (step 178);

performing another 16-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY (step 180);

performing a bit-wise exclusive or of PREV_OUTPUT with a 32-bit unsigned long integer of the current state of the object key provided by an incremented index into the current state of the object key (I_LONG_INT_OBJECT_KEY) (step 182);

rotating PREV_OUTPUT to the right I_BYTE_OBJECT_KEY modulus 31 plus 1 times (step 184);

performing a 32-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY (step 186);

repeating the previous set of operations eighty-four times substituting the seed unsigned byte with a byte from the four byte output block provided by the previous set of operations recursively setting the current output block to the next output block when the current output block is exhausted and utilizing a different ordered byte each round (step 188);

performing a byte transposition of the bytes in the new 256 byte output block (N_OUTPUT) provided by the previous set of operations utilizing the following operation:

performing a byte-wise index through N_OUTPUT and switching the current byte of N_OUTPUT with the N_OUTPUT byte indexed at position I_BYTE_OBJECT_KEY and

finally indexing through the entire block of N_OUTPUT (step 190). This modification method is used to modify the object key for each block of input data. Accordingly, the object key for creating the digital signature is also is dynamic as earlier described.

In the preferred embodiment, the encrypted ciphertext is signed by the originator to authenticate the information. More specifically, a digital signature is generated which is appended to the ciphertext. In the present invention, the digital signature is created under the control of the key, preferably an object key comprising data and methods that modify the data, and is unique to each ciphertext file by using the ciphertext as input into the digital signature generation process.

Accordingly, the present invention overcomes the disadvantages of known encryption processes by creating ciphertext immune to currently available cryptanalytic attacks. The cryptographic communications system of the present invention provides for an encryption process utilizing a dynamic object key. The initial state of the object key is created by the user and a method that modifies the keys based on seeding from the random session object key which is also a dynamic key whose initial state is set by a random number. Based on the object key, a different key schedule is used to encrypt each block of data in the preferred block cipher data encryption process. It will be appreciated by those skilled in the art that the use of a dynamic object key and encrypting with different key schedules can make most currently available encrypting processes stronger, i.e., more immune to cryptanalytic attacks. Tho authenticate the ciphertext, a 2048-bit secret key digital signature is appended to the ciphertext. The digital signature is generated using the ciphertext as input and is unique for

each ciphertext file. More specifically, the ciphertext is sent into a one-way keyed hash function to produce the digital signature.

Although the illustrative embodiments of the present invention have been described herein with reference to the accompanying drawings, it is to be understood that the invention is not limited to those precise embodiments, and that various other changes and modifications may be effected therein by one skilled in that art without departing from the scope of the invention.

What is claimed is:

1. A computer implemented method for encrypting data comprising the steps of:

creating an object key comprising data and methods that operate on said data; and

encrypting input plaintext data utilizing said object key in conjunction with an encryption process.

2. A computer implemented method as defined in Claim 1, wherein the encryption process comprises a block cipher system such that blocks of data bits are encrypted.

3. A computer implemented method as defined in Claim 2, wherein the object key is dynamic and changes with each data block encrypted.

4. A computer implemented method as defined in Claim 1, wherein the object key is dynamic.

5. A computer implemented method as defined in Claim 1, further comprising prior to the step of encrypting, the steps of:

creating an initial state of the object by the user;

creating an initial state of a random session object key;

encrypting the initial state of the random session object key in a block cipher encryption process with the initial state of the object key; and

modifying the object key based on seeding from the random session object key before each input data block so that each block is encrypted based on a different object key.

6. A computer implemented method as defined in Claim 5, wherein the initial state of the random session object key is created by generating a random number.

7. A computer implemented method as defined in Claim 6, wherein a new random number is generated and assigned as the initial state of the random session object key each time the block cipher encryption process is executed.

8. A computer implemented method as defined in Claim 5, wherein each object key is associated with a different key schedule for encrypting each input data block with said different key schedule.

9. A computer implemented method as defined in Claim 4, wherein the method of modifying the dynamic object key comprises the steps of:

generating a random seed unsigned byte and bit wise exclusive or to an unsigned byte of a current state of the object key provided by an incremented index into the current state of the object key (I_BYTE_OBJECT_KEY);

performing an unsigned byte addition on the output byte of the previous operation (PREV_OUTPUT) with I_BYTE_OBJECT_KEY;

performing a 16-bit multiplication of PREV_OUTPUT and I_BYTE_OBJECT_KEY modulus 254 and add 2;

performing a 16-bit addition of PREV_OUTPUT and I_BYTE_OBJECT_KEY;

performing another 16-bit addition of PREV_OUTPUT and I_BYTE_OBJECT_KEY;

performing a bit-wise exclusive or of PREV_OUTPUT with a 16-bit unsigned integer of the current state of the object key provided by an incremented index into the current state of the object key (I_INT_OBJECT_KEY);

rotating PREV_OUTPUT to the right I_BYTE_OBJECT_KEY modulus 15 plus 1 times;

performing a 16-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY;

performing a 16-bit multiplication of PREV_OUTPUT and I_INT_OBJECT_KEY with the lower order byte of I_INT_OBJECT_KEY modulus 254 plus 2;

performing a 16-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY;

performing another 16-bit addition of PREV_OUTPUT and I_INT_OBJECT_KEY;

performing a bit-wise exclusive or of PREV_OUTPUT with a 32-bit unsigned long integer of the current state of the object key provided by an incremented index into the current state of the object key (I_LONG_INT_OBJECT_KEY);

rotating PREV_OUTPUT to the left I_BYTE_OBJECT_KEY modulus 31 plus 1 times;

performing a bit-wise exclusive or of PREV_OUTPUT with I_LONG_INT_OBJECT_KEY;

repeating the previous set of operations eighty-four times substituting the random seed unsigned byte with a byte from a four byte output block provided by the previous set of operations recursively setting the current output block to a next output block when the current output block is exhausted, utilizing a different ordered byte each round;

performing a byte transposition of the bytes in the new 256 byte output block (N_OUTPUT) provided by the previous set of operations utilizing the following steps:

performing a byte-wise index through N_OUTPUT; switching the current byte of N_OUTPUT with the N_OUTPUT byte indexed at position I_BYTE_OBJECT_KEY; and indexing through the entire block of N_OUTPUT.

10. A computer implemented method as defined in Claim 1, wherein said object key is dynamic and a modification method of said object key includes a hashing function.

11. A computer implemented method as defined in Claim 1, wherein the object key is dynamic and includes at least two sub-object keys, and further wherein each sub-object key has a unique modification method associated therewith.

12. A computer implemented method as defined in Claim 5, wherein the object key includes at least two sub-object keys and the random session object key operations with the object key are performed with only one of said sub-object keys.

13. A computer implemented method as defined in Claim 5, wherein creating the initial state of the random session key object comprises the steps of:

accessing a running time clock in a computer;

multiplying together unique byte elements of the object key and summing and performing a bit-wise exclusive or to the time clock;

using the output of the previous step as a seed for a rand() function available in C libraries;

using an output of the rand() function modulus 255 plus an offset of 1 plus the lower eight bits of a high resolution computer clock timer is calculated and stored as one byte of the initial state of the random session object key;

repeating the previous set steps for each byte in the initial state of the random session key object.

14. A computer implemented method as defined in Claim 5, wherein the modification method for the random session object key comprises the steps of:

indexing through each byte of the current state of the random session key object (I_BYTE_R_OBJECT) and replacing that byte with the output of the following operation:

double indexing into the object key with I_BYTE_R_OBJECT as a starting index and add an offset and I_BYTE_OBJECT_KEY;

15. A computer implemented method as defined in Claim 17, wherein the modification method of said random session object key includes a hashing function.

16. A computer implemented method as defined in Claim 5, wherein said object key is first initialized with the random session key object by using an initial current state of the random session key object to provide a key schedule in the modification method of the object key.

17. A computer implemented method as defined in Claim 2, wherein input plaintext is compressed using a redundant byte reducing method and padded with random bytes to produce a file with a length that is evenly divisible by the block length so that the plaintext blocks are processed by said block cipher system.

18. A computer implemented method as defined in Claim 5, further including the step of performing a keyed transposition of ciphertext bytes after all input blocks are encrypted.

19. A computer implemented method as defined in Claim 2, wherein the encrypting step comprises the steps of:

transposition a substitution array whose elements contain unique numbers in reference to substitution array by switching a position of each element with a position provided by an element of a key.

20. A computer implemented method as defined in Claim 19, wherein the position provided by an element of the key is bounded by the size of said substitution array.

21. A computer implemented method as defined by Claim 2, wherein the block cipher encryption process comprises the steps of:

transpositioning a substitution array whose elements contain unique numbers in reference to said substitution array by switching a position of each element with a position provided by an element of a key, which position provided by an element of the key is bounded by the size of said substitution array which is composed of 256 elements;

transpositioning a traverse array whose elements contain unique numbers in reference to said transverse array by switching a position of each element with a position provided by an element of the key, the position provided by an element of the key is bounded by the size of the transverse array which is equal to the block size;

replacing each input byte transverse number of times with the value of the substitution array indexed with the input byte;

summing each output byte of the previous three steps to an element of the key to create ciphertext;

grouping the ciphertext in a 32-bit sliding window and rotating to the left an element of the key modulus 31 plus 1 times, the window sliding by one byte after each rotation and this step being performed on all ciphertext bytes;

performing a bit-wise exclusive or of each cipher text byte to an element of the key;

transpositioning the substitution arry by switching a position of each element with a position provided by an element of the key, the position provided by an element of the key is bounded by a size of said substitution array;

transpositioning the transverse array by switching a position of each element with a position provided by an element of the key, the position provided by an element of the key is bounded by a size of said transverse array;

replacing each input byte transverse number of time with a value of the substitution array indexed with an input byte;

transpositioning the ciphertext by switching a position of each ciphertext element with a position provided by an element of the key, the position provided by an element of the key is bounded by a size of the block;

repeating the previous seven steps four times with the key elements being unique each time the key is accessed;

transpositioning each bit in the ciphertext block by switching a position of each ciphertext bit with a position provided by elements of the key, the position provided by elements of the key are bounded by the size of the blocks times eight; and

repeating the previous nine steps four times with the key elements being unique each time the key is accessed.

22. A computer implemented method as defined in Claim 21, wherein said key is the object key.

23. A computer implemented method as defined in Claim 21, wherein the last transpositioning step uses a switch key comprised of the following steps:

initializing the switch key with elements of an initial state of the object key;

grouping the switch key by 32-bit blocks;

replacing the current switch key element with the following process:

performing a bit-wise exclusive or of the current switch key element to a switch key element indixed two elements from the current element;

rotating the output of the previous step to the right switch key indexed three elements from the current element modulus thirty-one plus one;

performing a bit-wise exclusive or of the output from the previous step to a switch key element indexed three elements from the current element;

repeating the previous three steps for each final transposition switch operation.

24. A computer implemented method as defined in Claim 23, wherein a hashing function is included in the creation of the switch key.

25. A computer implemented method for authenticating ciphertext, comprising the steps of:

generating a digital signature of a user using the ciphertext as input into a keyed one-way hash function; and

appending the digital signature to the input ciphertext.

26. A computer implemented method as defined in Claim 25, further comprising the steps of:

executing verification software which includes a user's secret digital signature key to regenerate the digital signature of the ciphertext; and

comparing the regenerated signature with the original signature appended to the ciphertext to determine whether the signatures are the same.

27. A computer implemented method as defined in Claim 25, wherein the step of generating includes the steps of:

dividing the input data into 256 byte data blocks;

further dividing the data block into 64 32-bit blocks (VAR_BLOCK);

modifying each VAR_BLOCK and element of a control key by a plurality of unique one-way irreversible hash functions;

repeating the previous step for all VAR_BLOCK - hash function combinations to create a running message digest;

modifying the running message digest using a bit-wise exclusive or to the next input data block;

repeating the previous four steps for all data blocks; and

transpositioning each byte of an output of the previous step by switching the position of each byte with another byte at a position provided by an element of the control key.

28. A computer implemented method as defined in Claim 25, wherein said keyed one-way hash function utilizes an object key.

29. A computer implemented method as defined in Claim 27, wherein the object key utilizes the output rounds of the one-way hash function to seed the object key's modification methods.

30. A cryptographic communications system comprising:

at least two networked computer systems linked by a communication channel; and

each computer system including a central processing unit and a memory storage device for executing a block cipher encryption/decryption process;

wherein the encryption process transforms an input plaintext message to a ciphertext message and the decryption process transforms the ciphertext message to the input plaintext message, the encryption/decryption process using a dynamic object key which changes with each block of input data, each object key being associated with a different key schedule to encrypt/decrypt the input plaintext/output ciphertext message.

31. A cryptographic communications system as defined in Claim 30, wherein the encryption/decryption process further includes the use of a random session object key having an initial state randomly generated by the computer system, and wherein the object key modifications are based on seeding from the random session object key.

32. A cryptographic communications system as defined in Claim 31, wherein an initial state of the object key is created by the user and wherein the initial state of the random session object key is created by the computer system generating a random number.

33. A cryptographic communications system as defined in Claim 30, wherein a digital signature is generated and appended to the ciphertext associated with each input plaintext message using a keyed one-way hash function utilizing an object key.

34. A cryptographic communications system as defined in Claim 30, wherein the block cipher encryption/decryption process includes use of a keyed transposition of a sequence of integers provides a coat of substitution rounds for a particular input entering an S-box.

35. A cryptographic communications system as defined in Claim 30, wherein a digital signature is generated for each ciphertext file by using the ciphertext as input into the digital signature generation process.

ABSTRACT OF THE DISCLOSURE

A computer implemented method and device for creating object keys to be used with a 4096-bit secret key block cipher data encryption process and a 2048-bit secret key digital signature process. The object keys are dynamic keys, i.e., changing throughout the encryption process. The dynamic object keys are composed of a static initial state that is created by the user and a method that modifies the keys based on seeding from a random session key object. The object key modification is performed for each plaintext data block so that each data block is encrypted using a different key. The initial state of the object key is also used in a block cipher encryption process to encrypt a 512-bit random session key. Data blocks of 64 bytes each are encrypted utilizing a different key, provided by the object key, for each block. The ciphertext (encrypted file) is transmitted into a keyed hashed function that utilizes a 2048-bit object key to produce a unique 2048-bit digital signature that is appended to the ciphertext. The digital signature object key is seeded with the input data. Decryption is accomplished by reversing the encryption process.

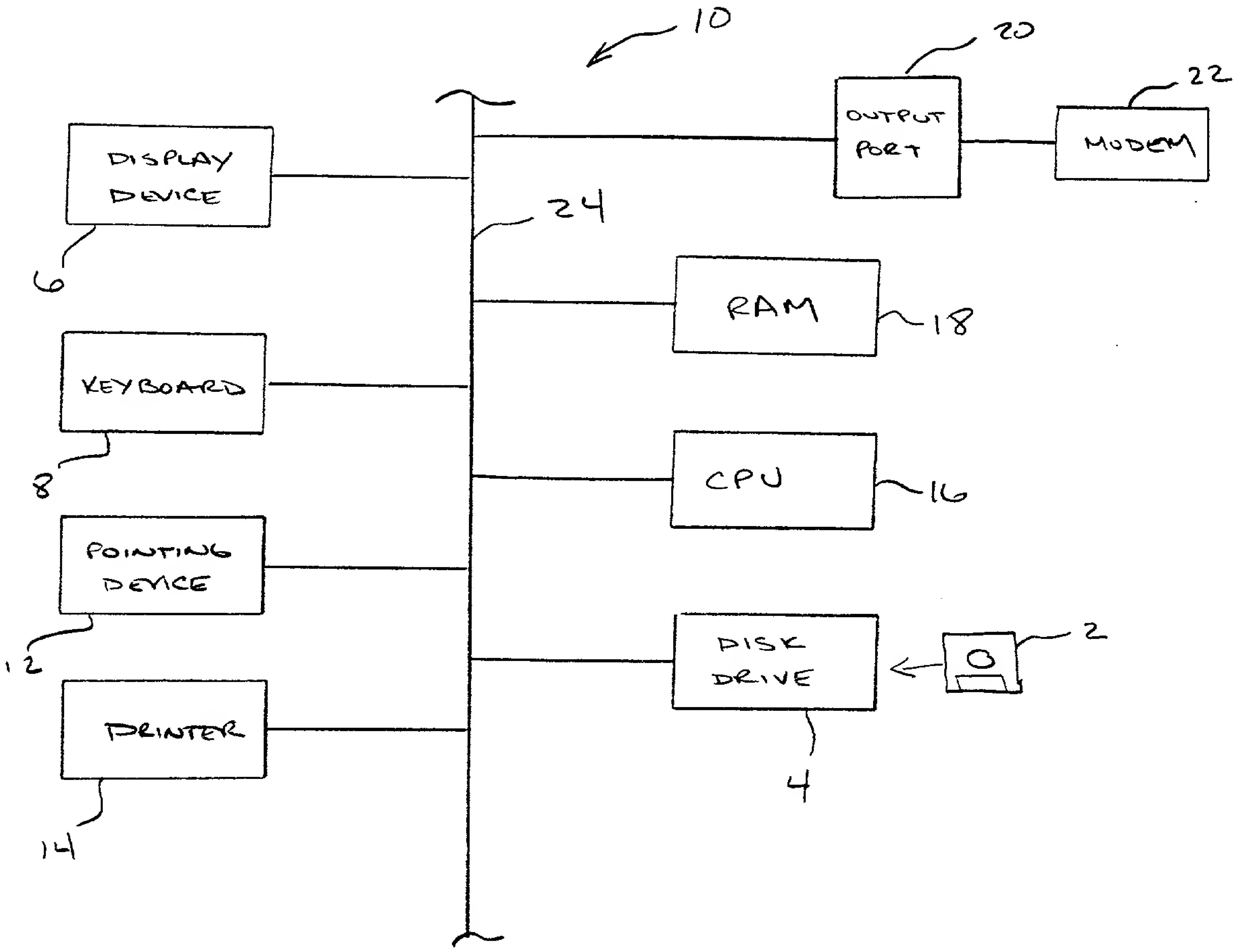

10
20
OUTPUT PORT
22
MODEM
DISPLAY DEVICE
6
24
RAM
18
KEYBOARD
8
CPU
16
POINTING DEVICE
12
DISK DRIVE
2
4
PRINTER
14

FIG. 1

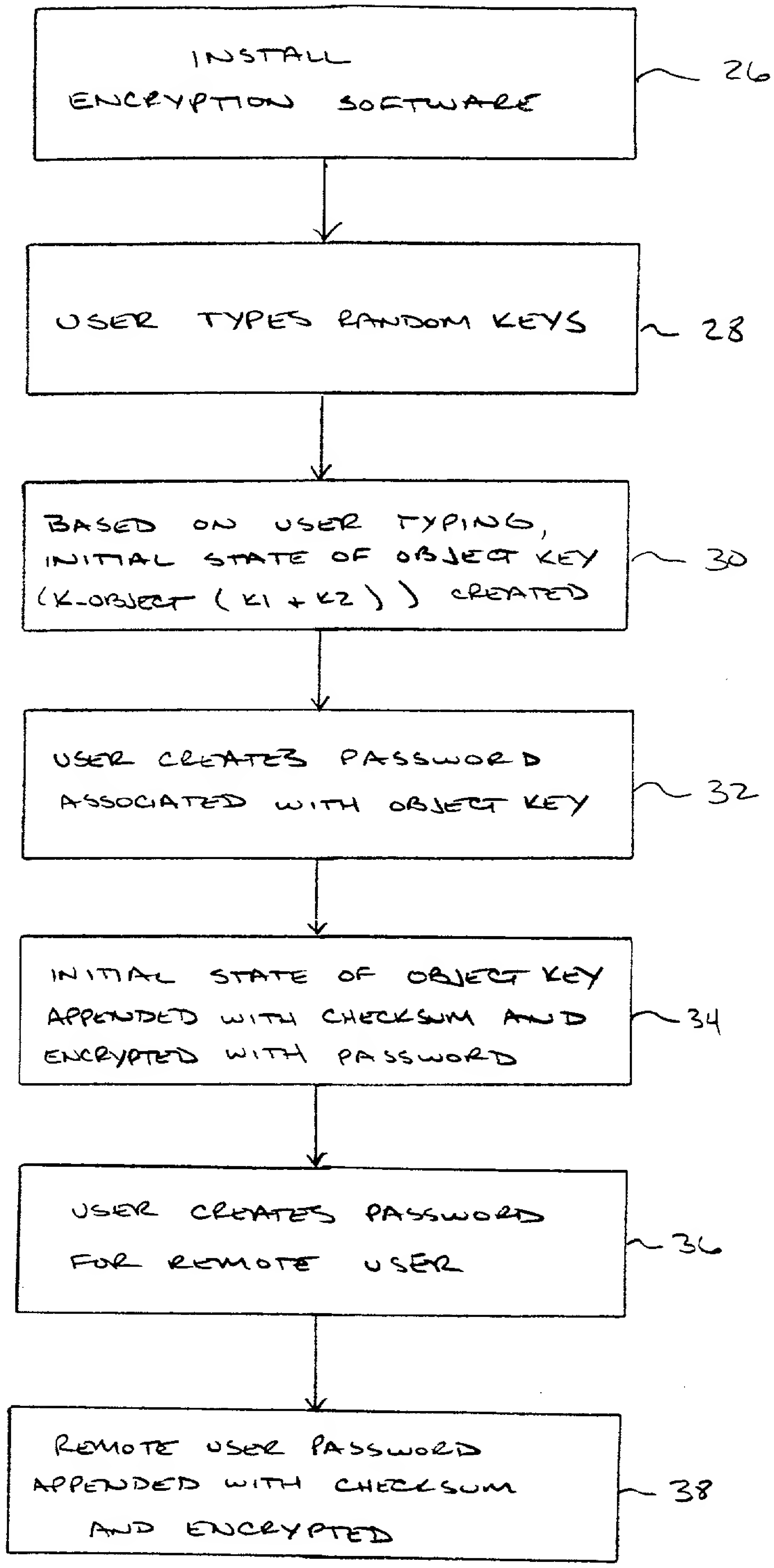
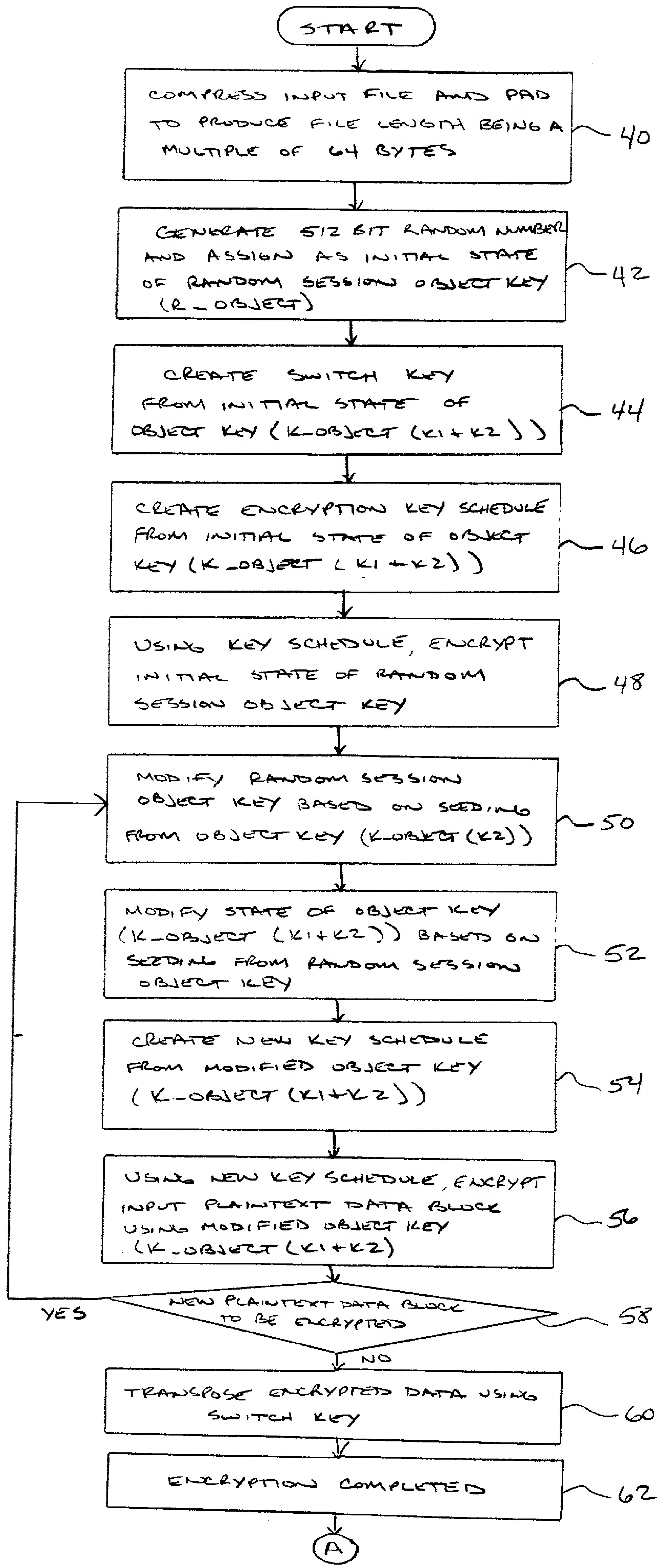

INSTALL ENCRYPTION SOFTWARE
26
USER TYPES RANDOM KEYS
28
BASED ON USER TYPING, INITIAL STATE OF OBJECT KEY (K-OBJECT (K1 + K2)) CREATED
30
USER CREATES PASSWORD ASSOCIATED WITH OBJECT KEY
32
INITIAL STATE OF OBJECT KEY APPENDED WITH CHECKSUM AND ENCRYPTED WITH PASSWORD
34
USER CREATES PASSWORD FOR REMOTE USER
36
REMOTE USER PASSWORD APPENDED WITH CHECKSUM AND ENCRYPTED
38

FIG. 2

FIG. 3


START
COMPRESS INPUT FILE AND PAD TO PRODUCE FILE LENGTH BEING A MULTIPLE OF 64 BYTES
40
GENERATE 512 BIT RANDOM NUMBER AND ASSIGN AS INITIAL STATE OF RANDOM SESSION OBJECT KEY (R_OBJECT)
42
CREATE SWITCH KEY FROM INITIAL STATE OF OBJECT KEY (K-OBJECT (K1+K2))
44
CREATE ENCRYPTION KEY SCHEDULE FROM INITIAL STATE OF OBJECT KEY (K_OBJECT (K1+K2))
46
USING KEY SCHEDULE, ENCRYPT INITIAL STATE OF RANDOM SESSION OBJECT KEY
48
MODIFY RANDOM SESSION OBJECT KEY BASED ON SEEDING FROM OBJECT KEY (K-OBJECT (K2))
50
MODIFY STATE OF OBJECT KEY (K_OBJECT (K1+K2)) BASED ON SEEDING FROM RANDOM SESSION OBJECT KEY
52
CREATE NEW KEY SCHEDULE FROM MODIFIED OBJECT KEY (K-OBJECT (K1+K2))
54
USING NEW KEY SCHEDULE, ENCRYPT INPUT PLAINTEXT DATA BLOCK USING MODIFIED OBJECT KEY (K_OBJECT (K1+K2)
56
NEW PLAINTEXT DATA BLOCK TO BE ENCRYPTED
58
YES
NO
TRANSPOSE ENCRYPTED DATA USING SWITCH KEY
60
ENCRYPTION COMPLETED
62
A

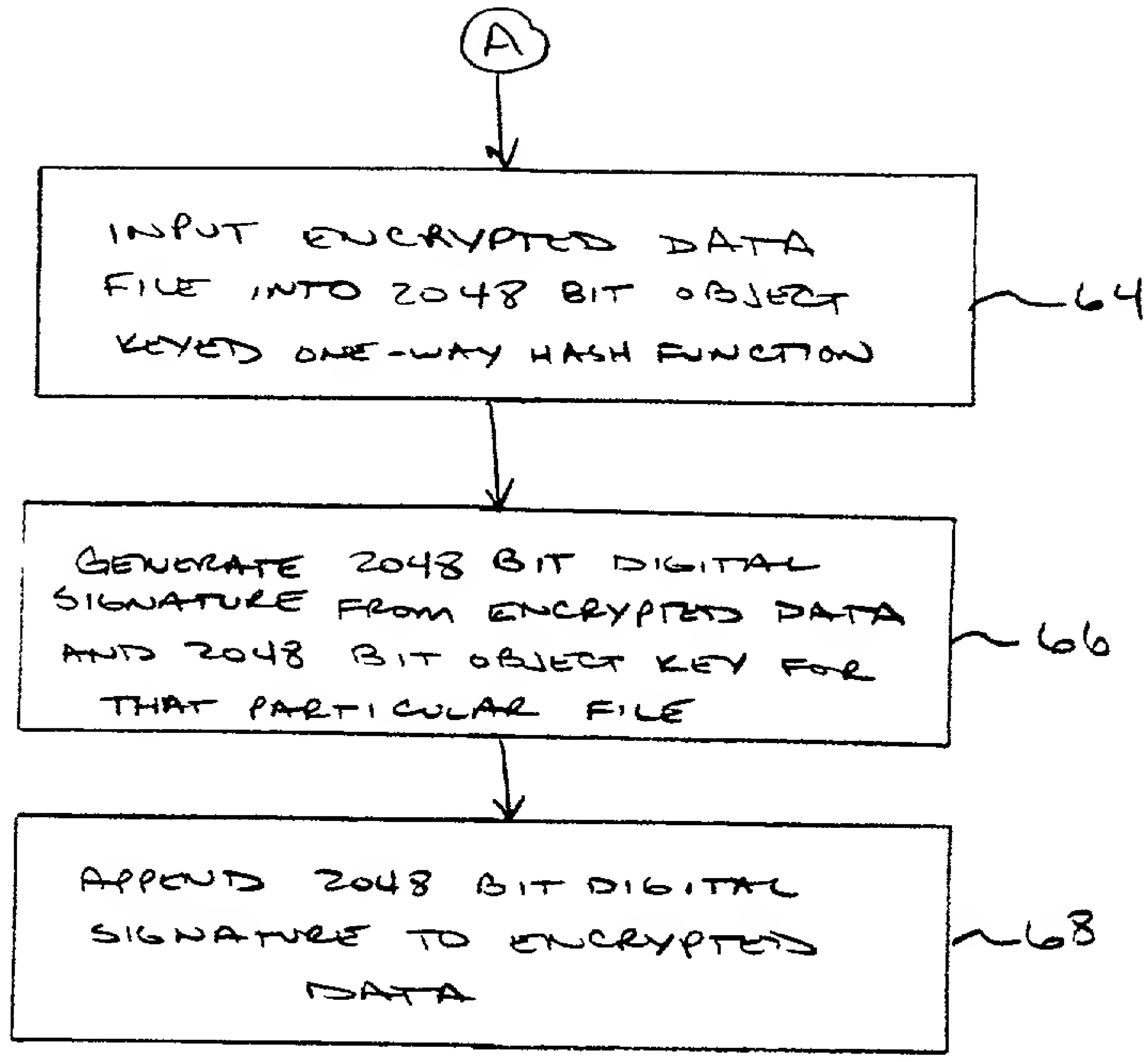

A
INPUT ENCRYPTED DATA FILE INTO 2048 BIT OBJECT KEYED ONE-WAY HASH FUNCTION
64
GENERATE 2048 BIT DIGITAL SIGNATURE FROM ENCRYPTED DATA AND 2048 BIT OBJECT KEY FOR THAT PARTICULAR FILE
66
APPEND 2048 BIT DIGITAL SIGNATURE TO ENCRYPTED DATA
68

FIG. 4

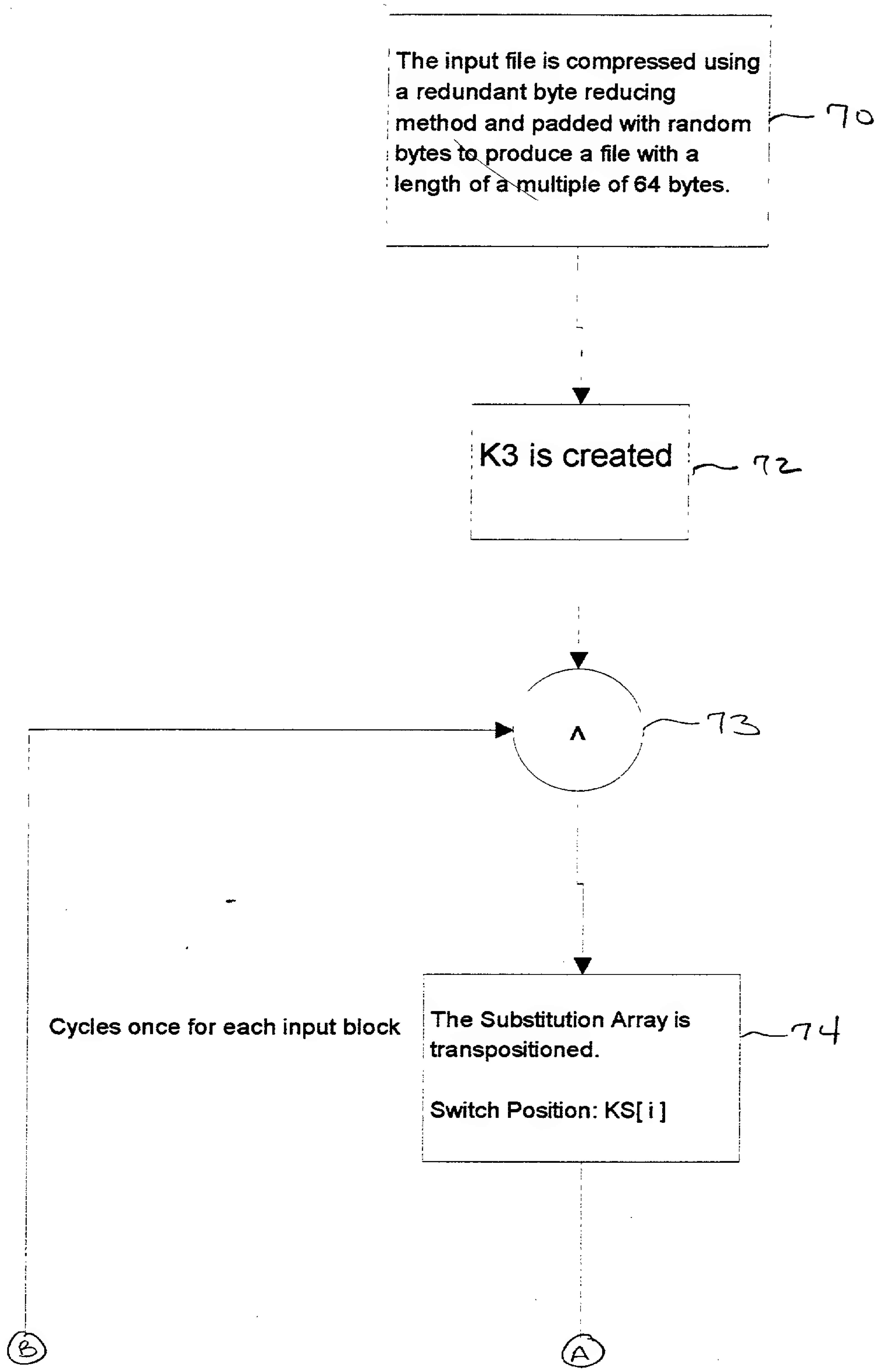
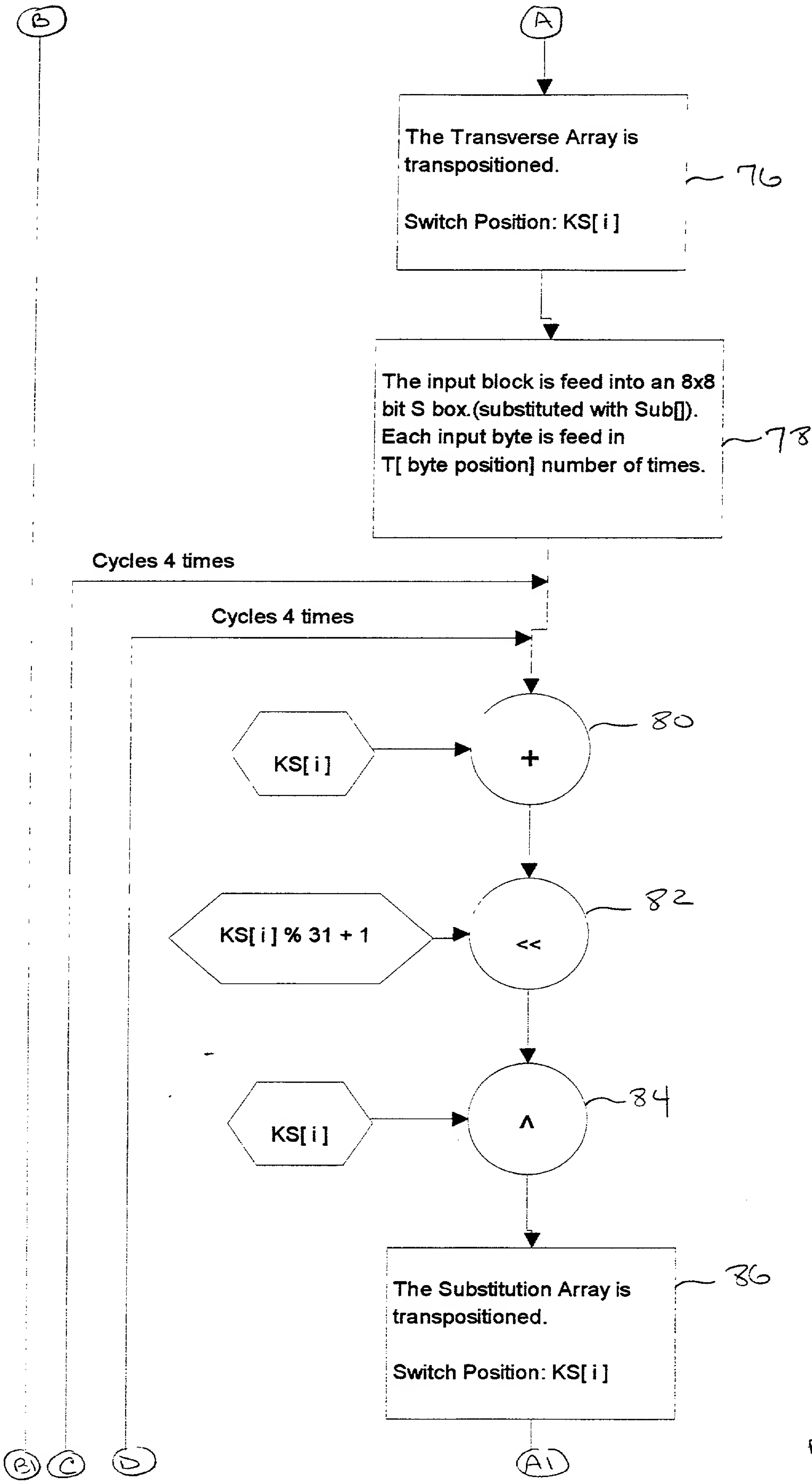

The input file is compressed using a redundant byte reducing method and padded with random bytes to produce a file with a length of a multiple of 64 bytes.
70
K3 is created
72
^
73
Cycles once for each input block
The Substitution Array is transpositioned.
Switch Position: KS[i]
74
B
A

FIG. 5



B
A
The Transverse Array is transpositioned.
Switch Position: KS[i]
76
The input block is feed into an 8x8 bit S box.(substituted with Sub[]). Each input byte is feed in T[byte position] number of times.
78
Cycles 4 times
Cycles 4 times
KS[i]
+
80
KS[i] % 31 + 1
<<
82
KS[i]
^
84
The Substitution Array is transpositioned.
Switch Position: KS[i]
86
B1
C
D
A1

FIG. 5 (CONT'D)

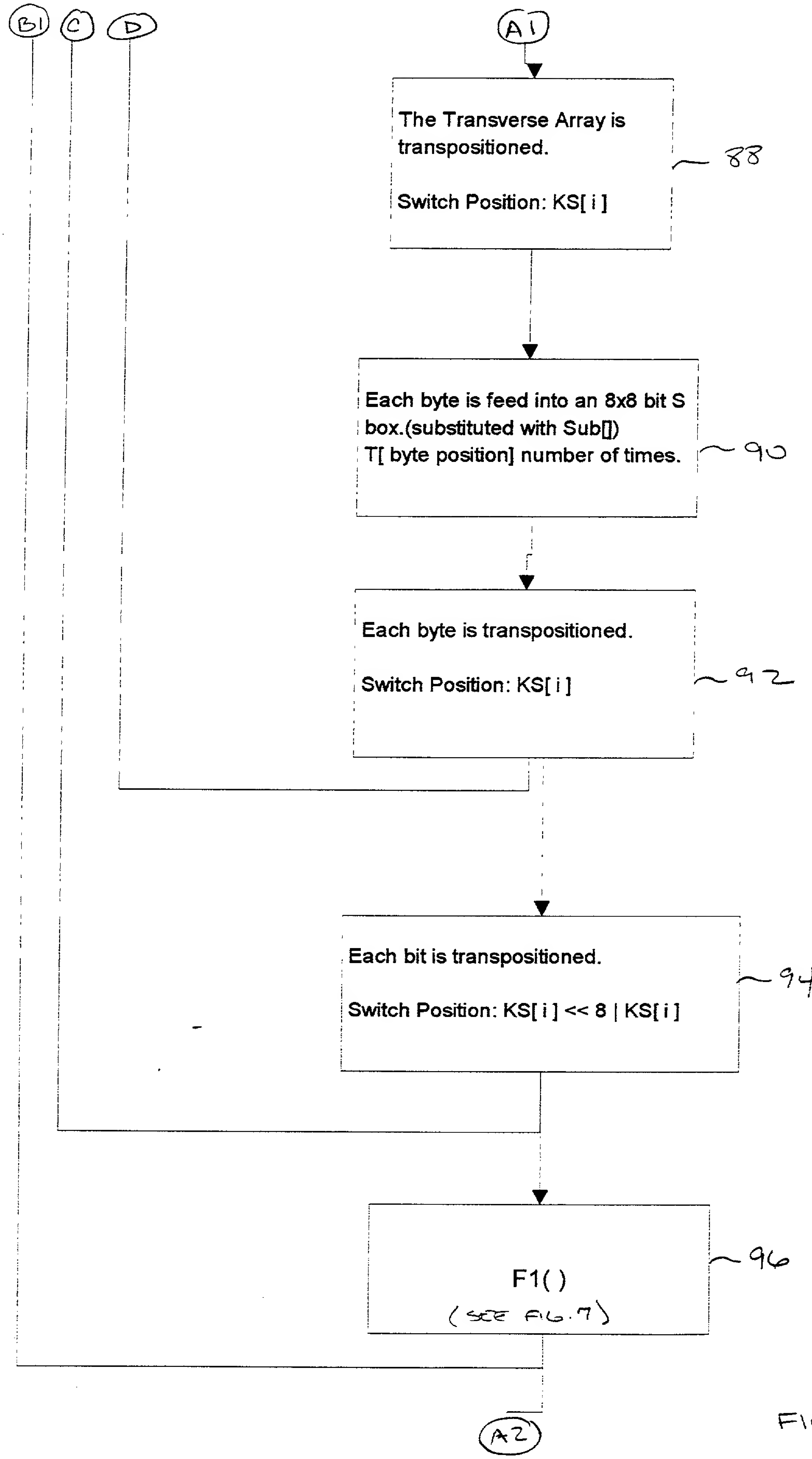

B1
C
D
A1
The Transverse Array is transpositioned.
Switch Position: KS[i]
88
Each byte is feed into an 8x8 bit S box.(substituted with Sub[])
T[byte position] number of times.
90
Each byte is transpositioned.
Switch Position: KS[i]
92
Each bit is transpositioned.
Switch Position: KS[i] << 8 | KS[i]
94
F1( )
(SEE FIG. 7)
96
A2

FIG. 5 (CONT'D)

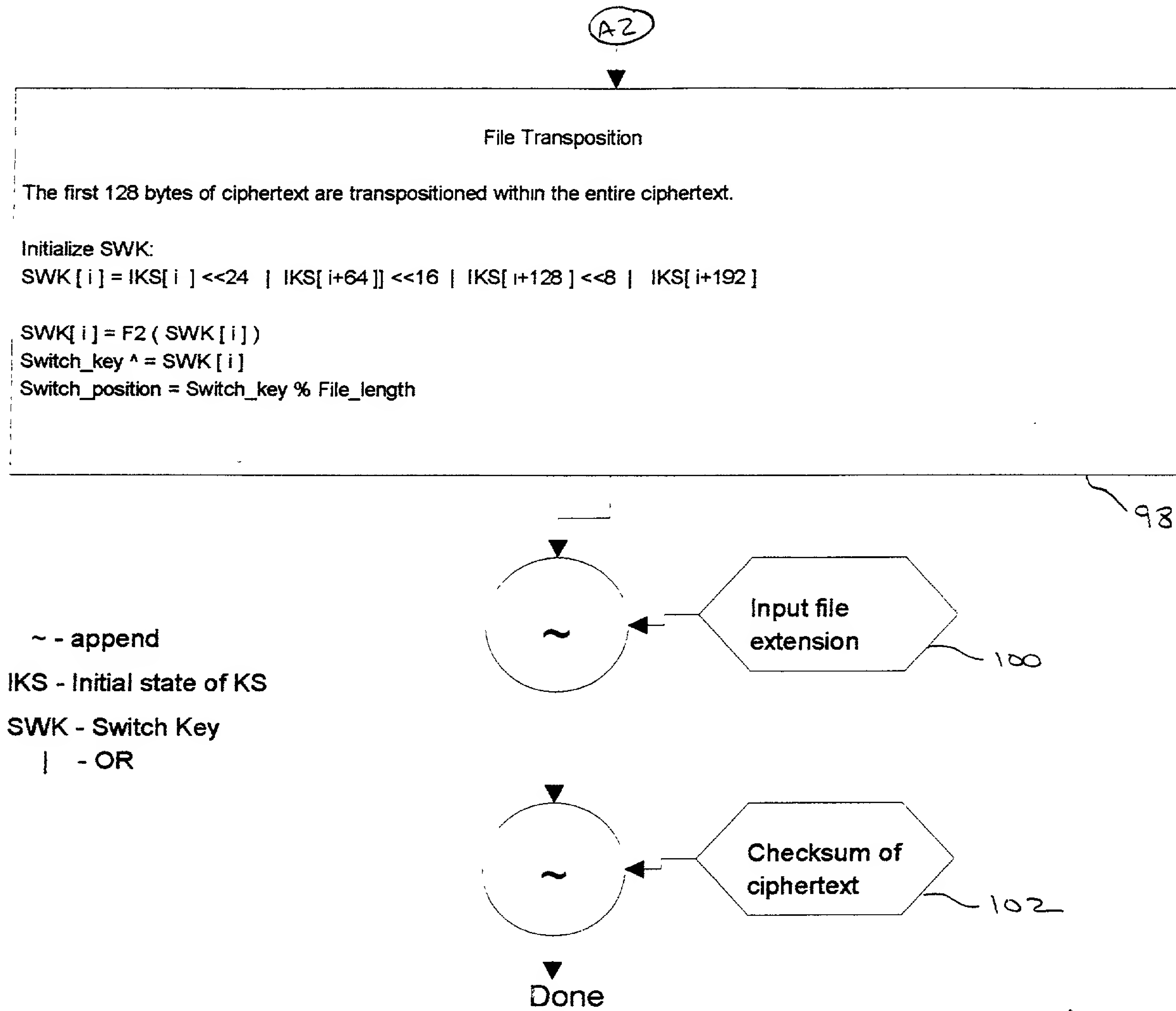


FIG. 5 (CONT'D)

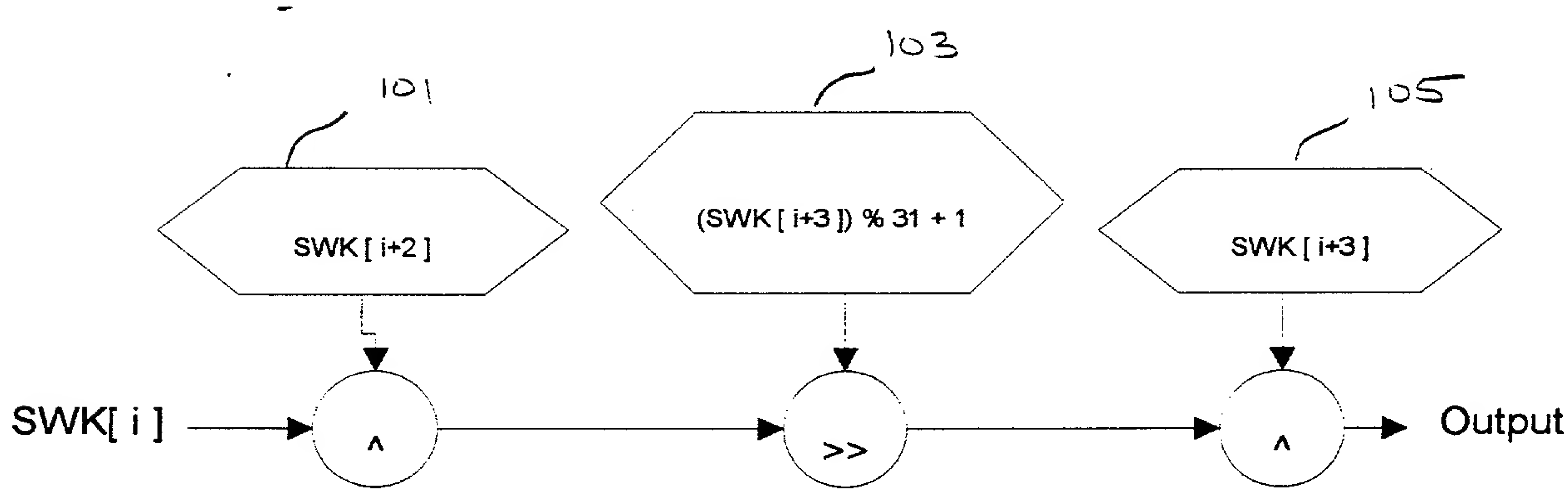


FIG. 6

K3 Modification

K3[i] + = (K2[K2[K3[i]]] % 255) + 113 + K2[i]

K1 Modification

K1_SEED ^ = K1[K1 [K3 [i]]]

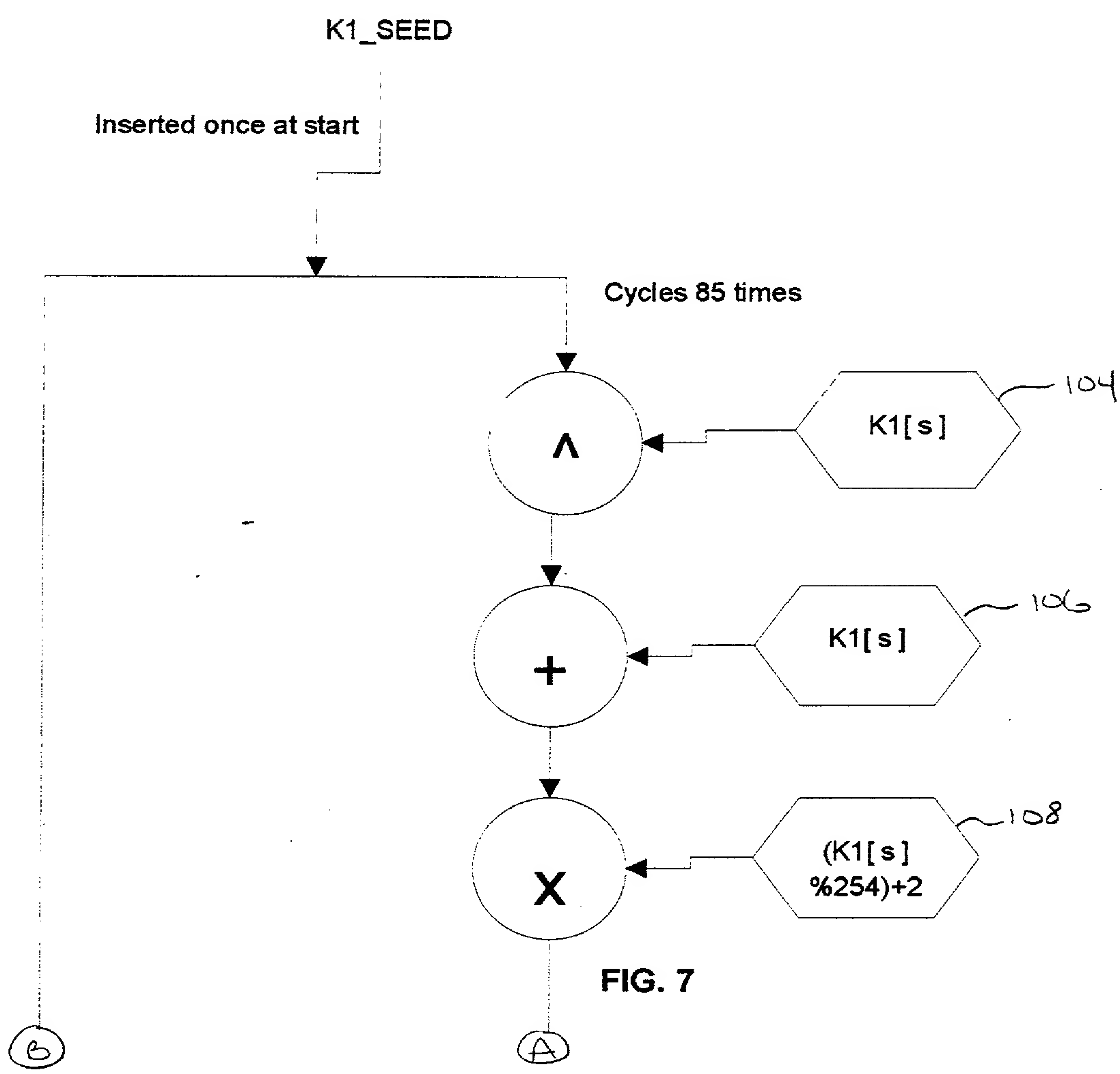


FIG. 7

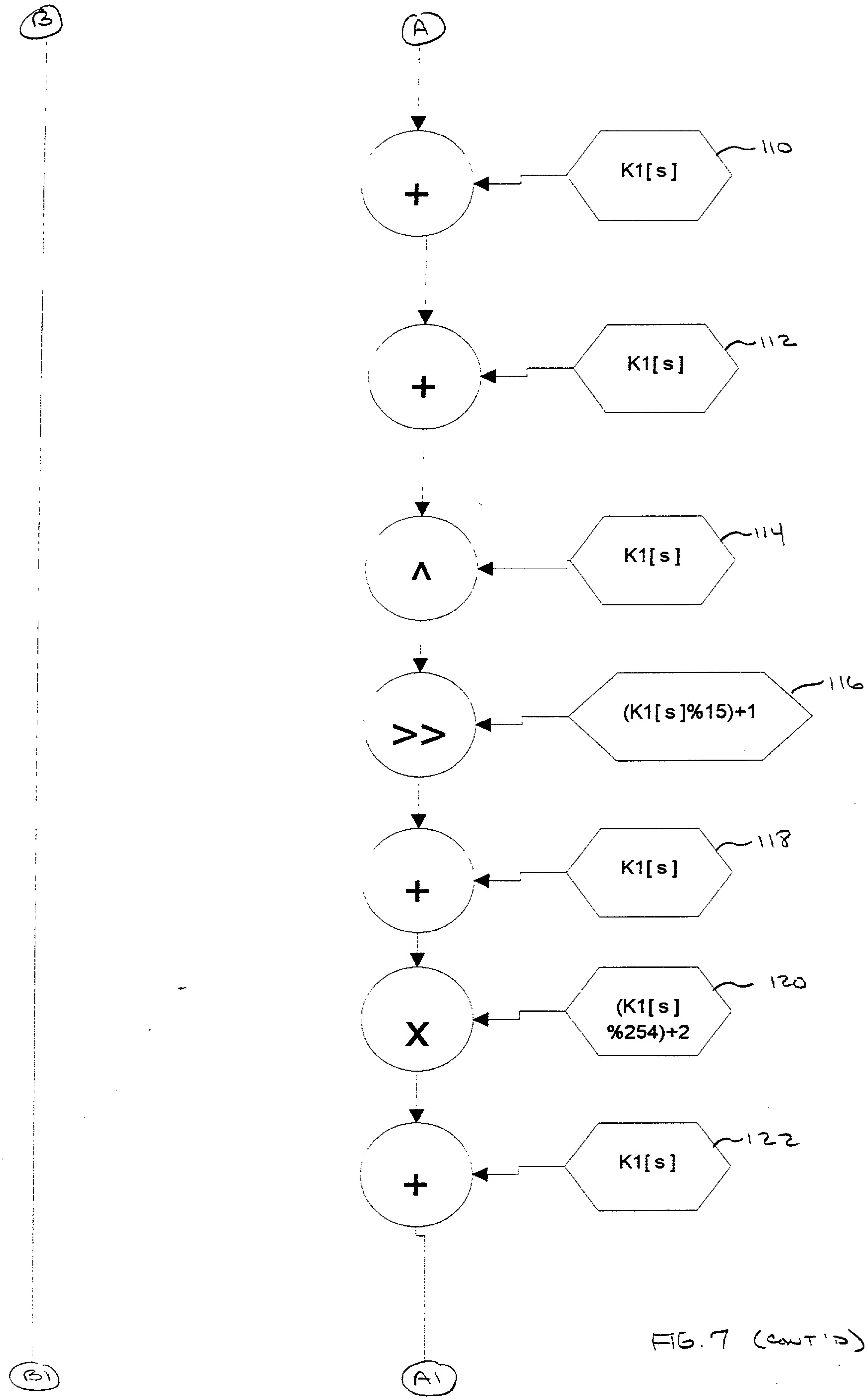

B
A
K1[s]
110
K1[s]
112
K1[s]
114
(K1[s]%15)+1
116
K1[s]
118
(K1[s]
%254)+2
120
K1[s]
122
B1
A1

FIG. 7 (CONT'D)

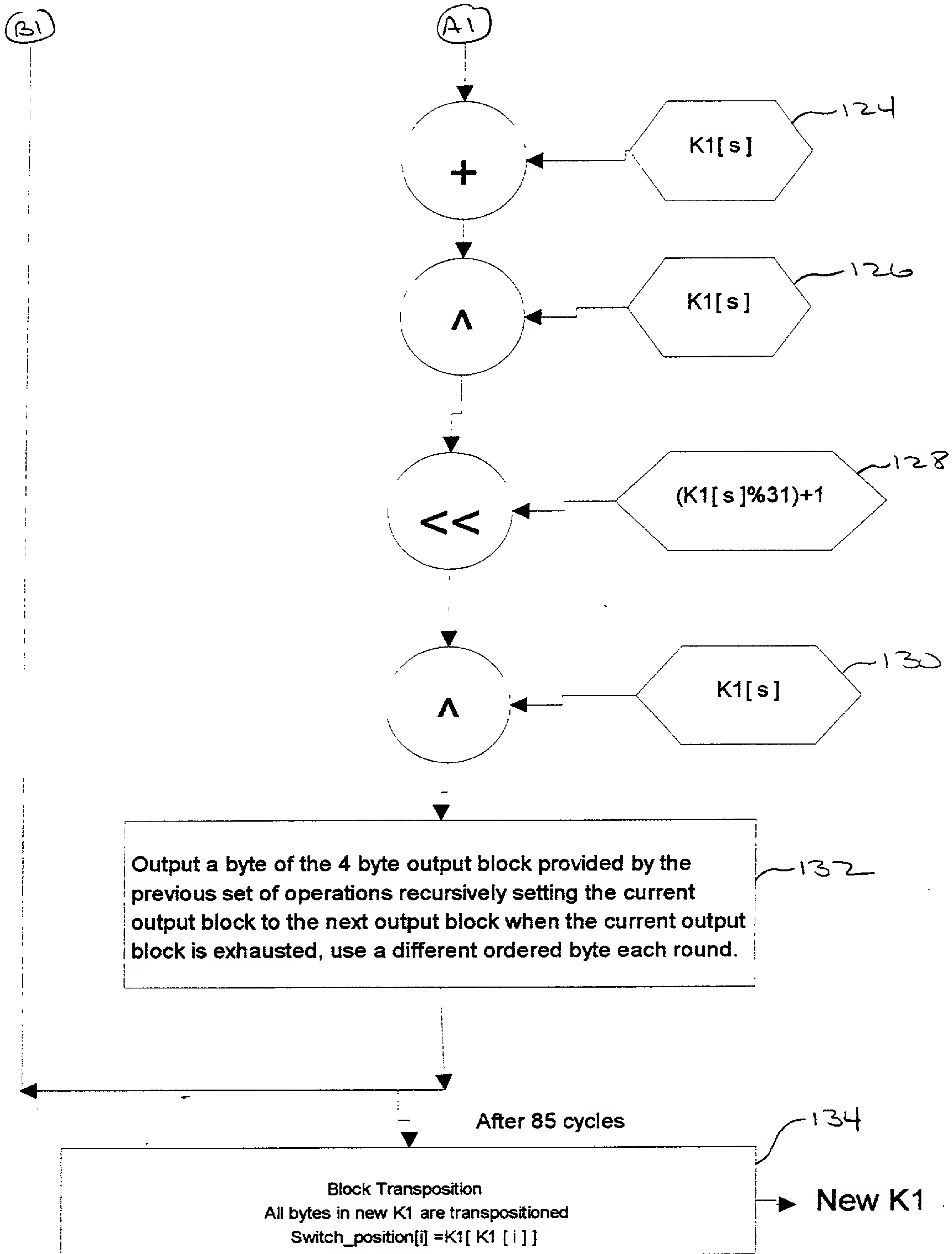

B1
A1
+
K1[s]
124
^
K1[s]
126
<<
(K1[s]%31)+1
128
^
K1[s]
130
Output a byte of the 4 byte output block provided by the previous set of operations recursively setting the current output block to the next output block when the current output block is exhausted, use a different ordered byte each round.
132
After 85 cycles
134
Block Transposition
All bytes in new K1 are transpositioned
Switch_position[i] =K1[K1 [i]]
New K1

FIG. 7 (CONT'D)

K2 Modification

K2_SEED + = (K3[K3 [#] % 64] % 253) + 3

K2_SEED ^ = K2 [K2 [K3 [K2[K3 [s % 64] + K2[#] % 192] % 64]]]

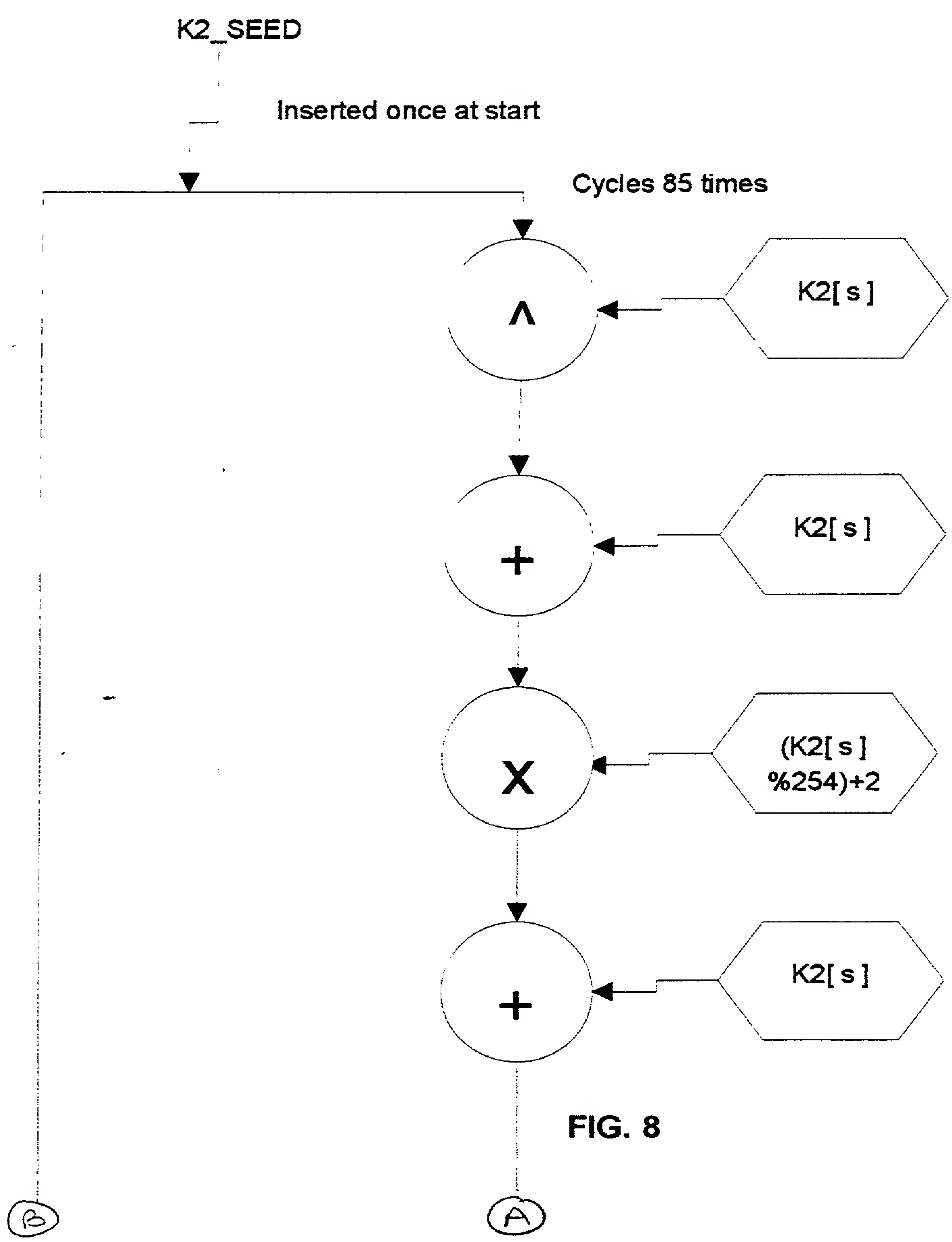


FIG. 8

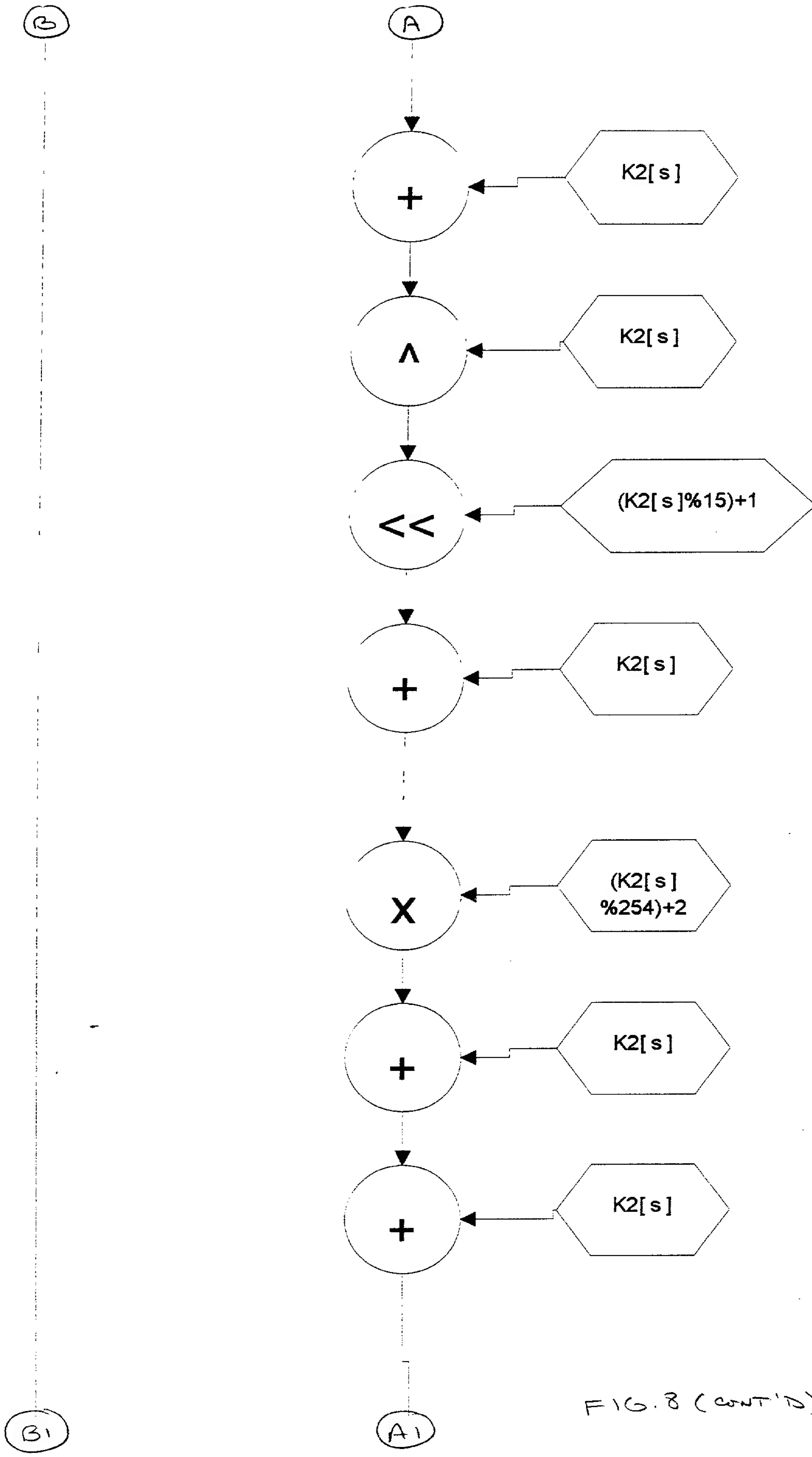


FIG. 8 (CONT'D)

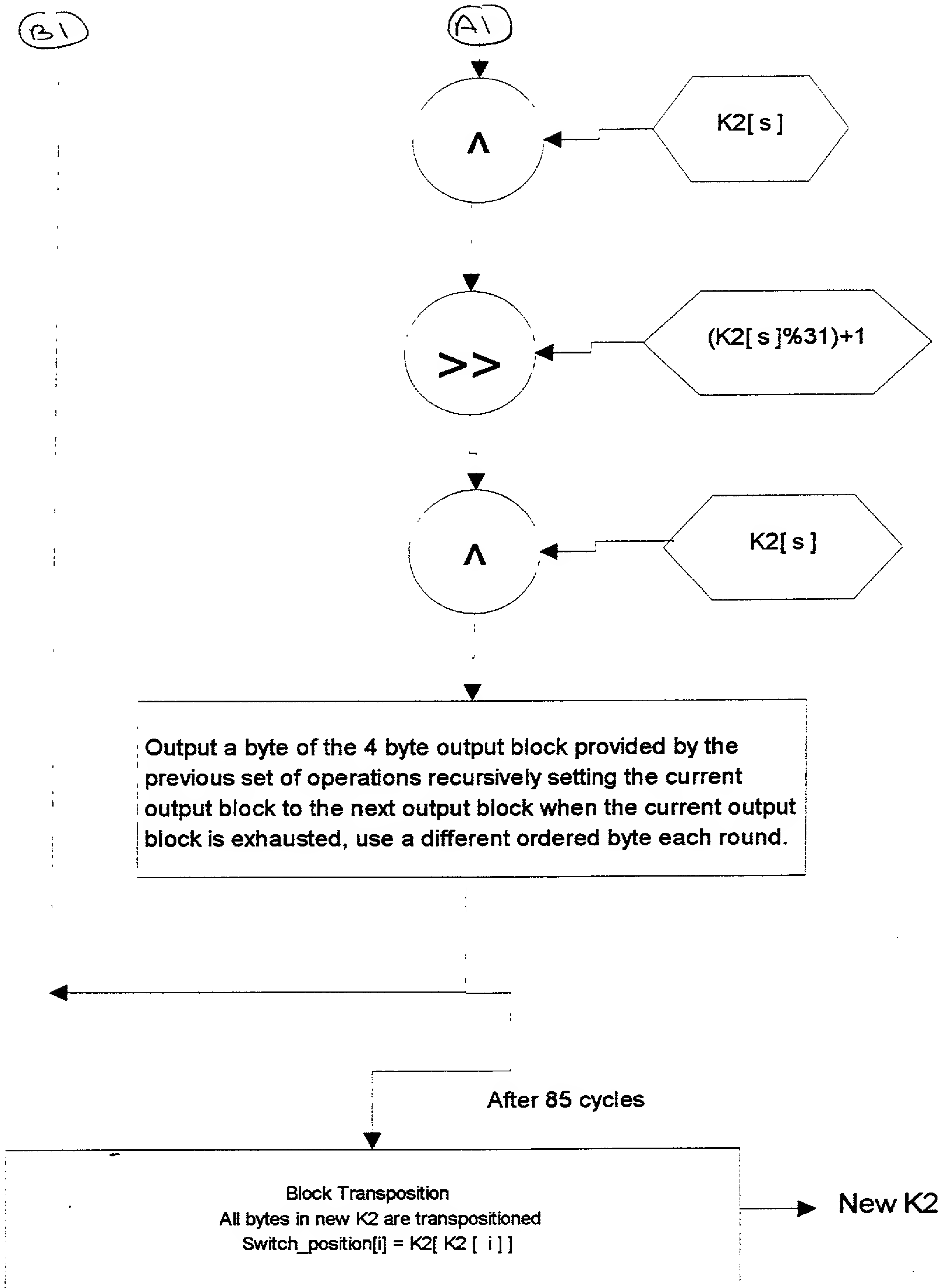

B1
A1
^
K2[s]
>>
(K2[s]%31)+1
^
K2[s]
Output a byte of the 4 byte output block provided by the previous set of operations recursively setting the current output block to the next output block when the current output block is exhausted, use a different ordered byte each round.
After 85 cycles
Block Transposition
All bytes in new K2 are transpositioned
Switch_position[i] = K2[K2 [i]]
New K2

FIG. 8 (CONT'D)

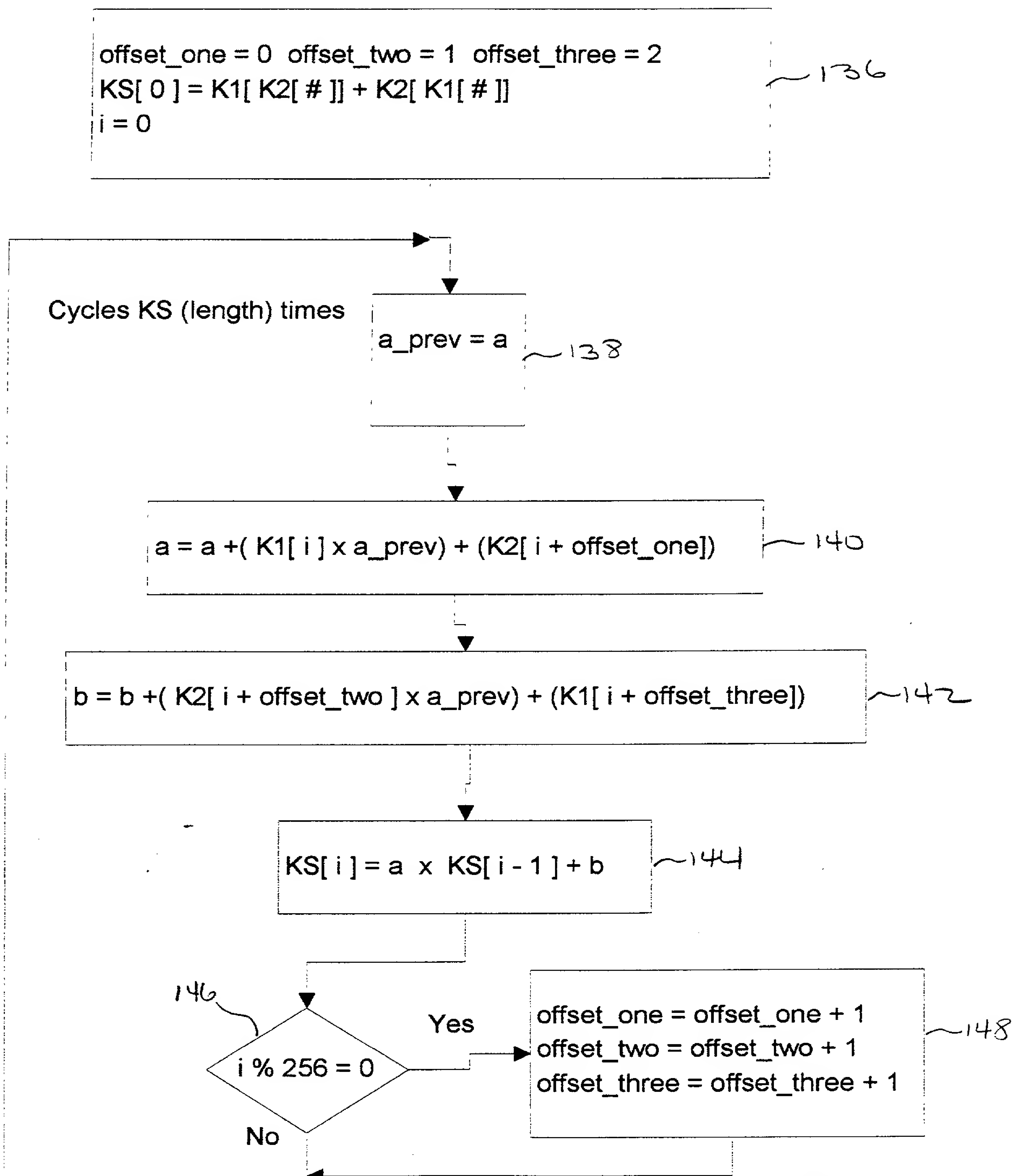

offset_one = 0 offset_two = 1 offset_three = 2
KS[0] = K1[K2[#]] + K2[K1[#]]
i = 0
136
Cycles KS (length) times
a_prev = a
138
a = a +(K1[i] x a_prev) + (K2[i + offset_one])
140
b = b +(K2[i + offset_two] x a_prev) + (K1[i + offset_three])
142
KS[i] = a x KS[i - 1] + b
144
146
i % 256 = 0
Yes
No
offset_one = offset_one + 1
offset_two = offset_two + 1
offset_three = offset_three + 1
148

FIG. 9

150

```
H1(v1,v2,v3,v4,v5,v6,v7) = ( v1 ^ v2 & v3 | ~v4 & v5 ^ v6 ^ v7)
H2(v1,v2,v3,v4,v5,v6,v7) = ( v1 & ~v2 ^ v3 ^ v4 ^ v5 & v6 | v7)
H3(v1,v2,v3,v4,v5,v6,v7) = ( v1 ^ v2 | v3 ^ v4 | ~v5 ^ v6 ^ ~v7)
H4(v1,v2,v3,v4,v5,v6,v7) = (~v1 ^ v2 & v3 | v4 ^ v5 ^ ~v6 & v7)
H5(v1,v2,v3,v4,v5,v6,v7) = ( v1 & v2 ^ v3 ^ ~v4 | v5 & v6 ^ v7)
H6(v1,v2,v3,v4,v5,v6,v7) = ( v1 ^ v2 & ~v3 | v4 & v5 | v6 ^ v7)
H7(v1,v2,v3,v4,v5,v6,v7) = ( v1 ^ v2 | v3 & v4 ^ v5 ^ ~v6 & v7)
H8(v1,v2,v3,v4,v5,v6,v7) = (~v1 & v2 ^ v3 | v4 ^ v5 & v6 ^ v7)

HASH(hnum,output,v1,v2,v3,v4,v5,v6,v7,key) = (output +=
key+hnum(v1,v2,v3,v4,v5,v6,v7)

HASH_FOR_KEY(hnum,result,output,v1,v2,v3,v4,v5,v6,v7,key) =
(result+=output+key+hnum(v1,v2,v3,v4,v5,v6,v7))
```

FIG. 10

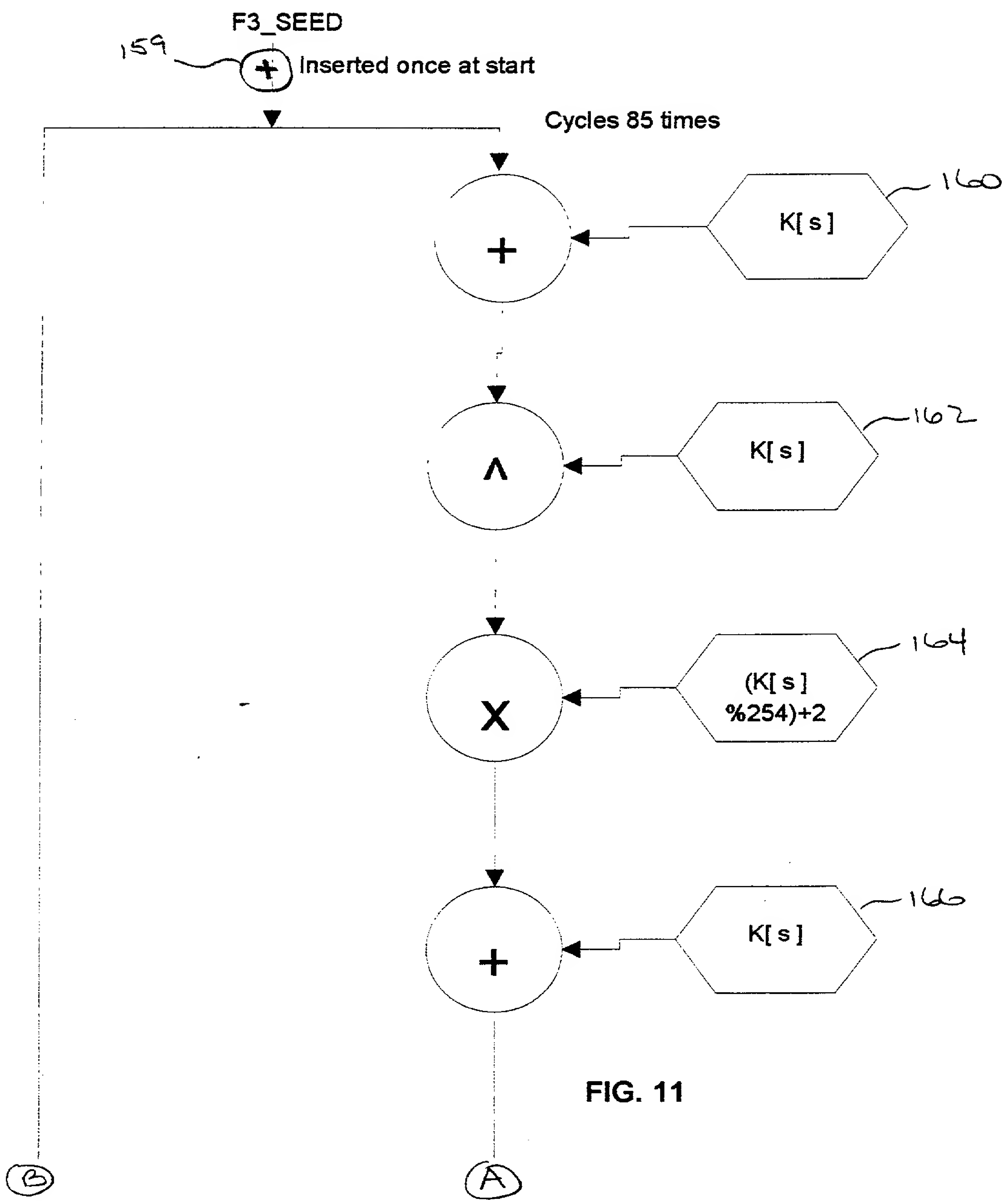

F3_SEED
159
+
Inserted once at start
Cycles 85 times
+
K[s]
160
^
K[s]
162
X
(K[s]
%254)+2
164
+
K[s]
166
B
A

FIG. 11

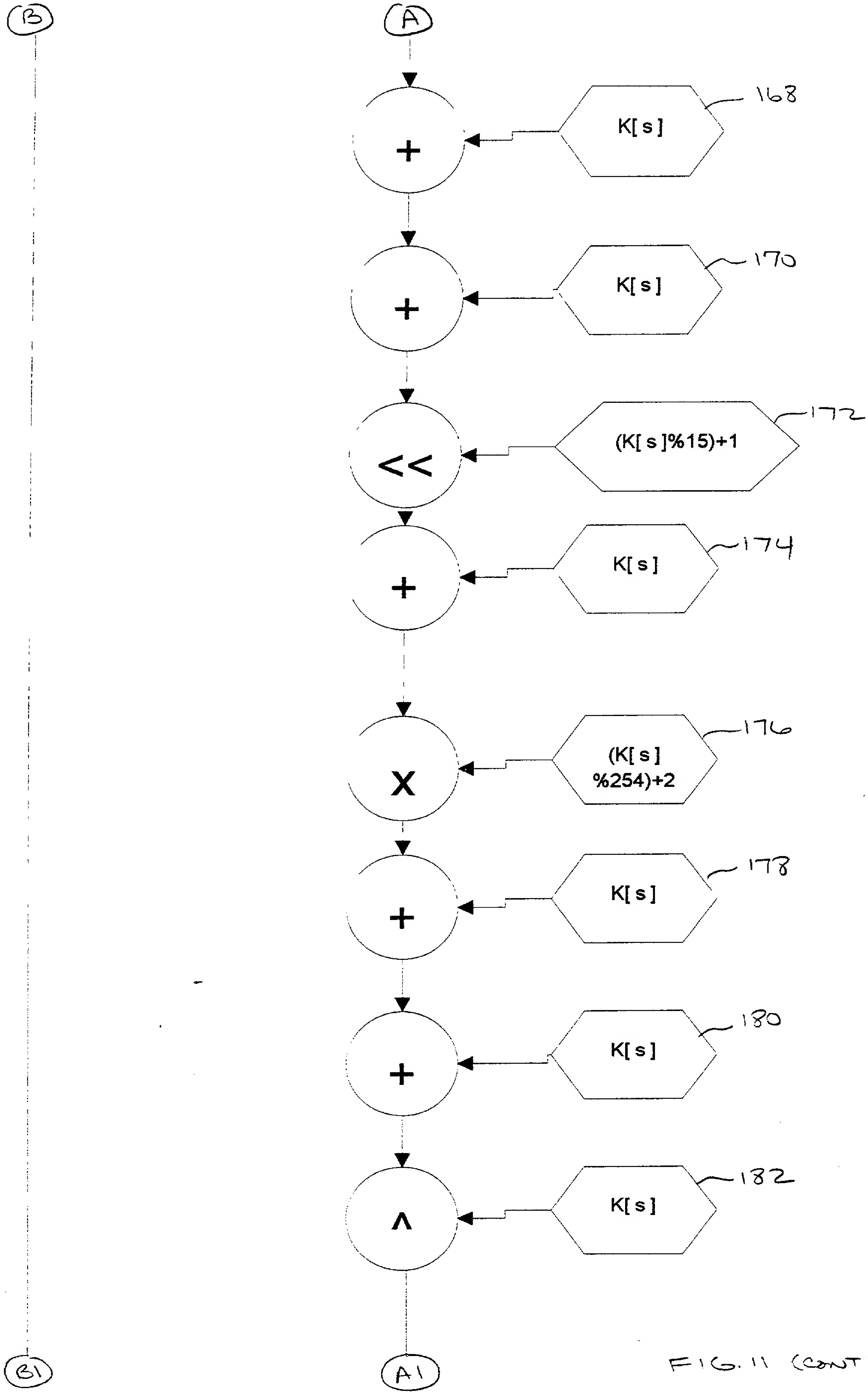
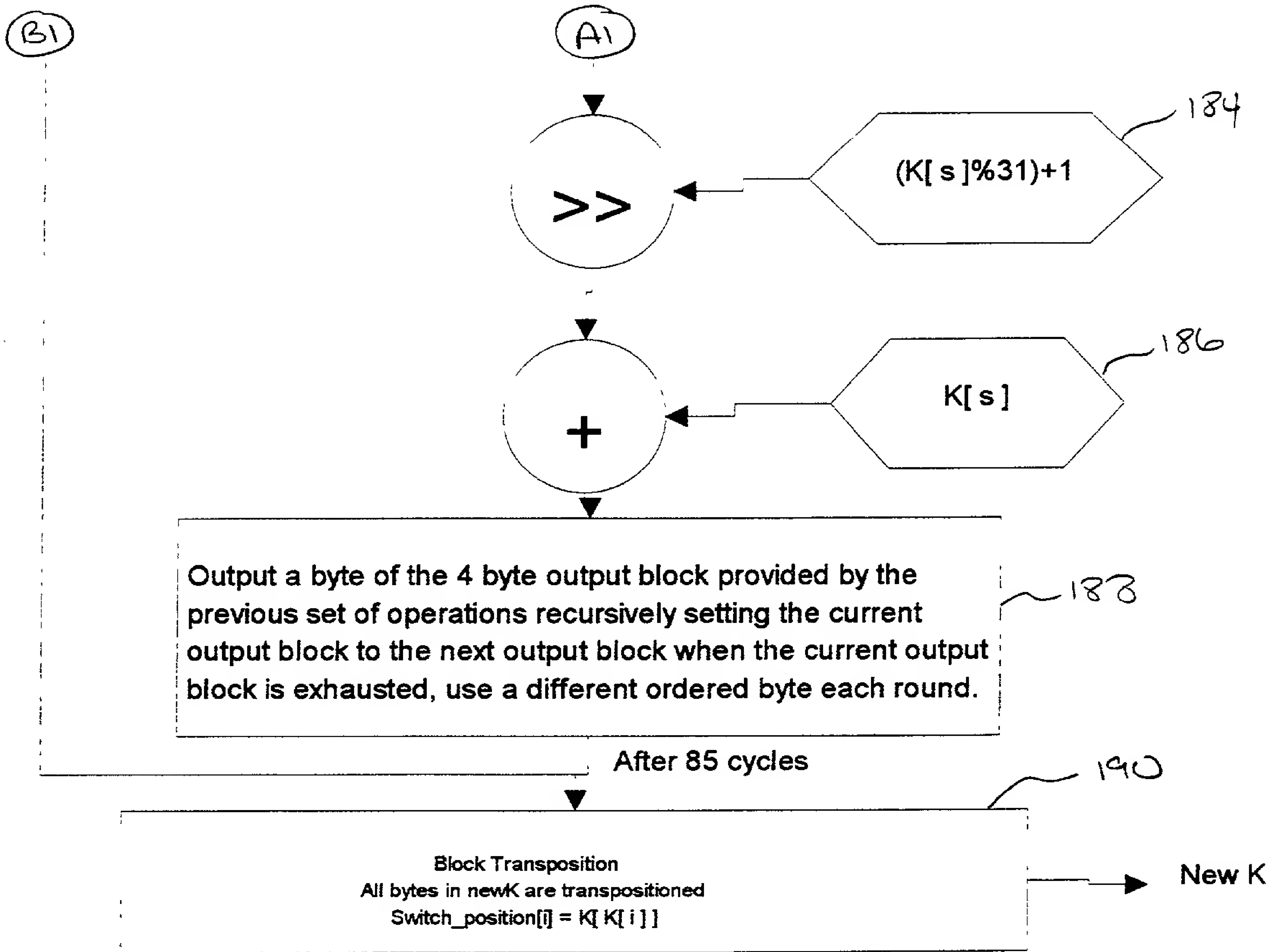

B
A
+
K[s]
168
+
K[s]
170
<<
(K[s]%15)+1
172
+
K[s]
174
X
(K[s]
%254)+2
176
+
K[s]
178
+
K[s]
180
^
K[s]
182
B1
A1

FIG. 11 (CONT'D)



input_block = 256 bytes of input, read from the input file.

var0 = 32 bit pointer assigned to input_block;
var1 = 32 bit pointer assigned to (input_block+32);
var2 = 32 bit pointer assigned to (input_block+64);
var3 = 32 bit pointer assigned to (input_block+96);
var4 = 32 bit pointer assigned to (input_block+128);
var5 = 32 bit pointer assigned to (input_block+160);
var6 = 32 bit pointer assigned to (input_block+192);
var7 = 32 bit pointer assigned to (input_block+224);

\# - static numbers
index++ - running index
rep - running index

for(rep=0;rep<8;rep++){ } - Code within "{ }" will be executed eight times and rep will be incremented after each loop.

FIG. 11 (CONT'D)

```
F3_SEED = (((K[(HASH_FOR_KEY(H1,o,K[#],K[#],K[#],K[#],K[#],K[#],K[#],K[#],K[(s)]))%64])>>
(HASH_FOR_KEY(H2,o,K[#],K[#],K[#],K[#],K[#],K[o%64],K[#],K[#],K[(s)]))%25));

F3(F3_SEED)

F3_SEED = (((K[(HASH_FOR_KEY(H1,o,K[#],K[#],K[#],K[o%64],K[#],K[#],K[#],K[#],K[(s)]))%64])>>
(HASH_FOR_KEY(H2,o,K[#],K[o%64],K[#],K[#],K[#],K[#],K[#],K[#],K[(s)]))%25));

F3(F3_SEED)

F3_SEED = (((K[(HASH_FOR_KEY(H1,o,K[o%64],K[#],K[#],K[#],K[#],K[#],K[#],K[#],K[(s)]))%64])>>
(HASH_FOR_KEY(H2,o,K[#],K[#],K[#],K[#],K[o%64],K[#],K[#],K[#],K[(s)]))%25));

F3(F3_SEED)
```

A

^

206

256 bytes of input is read and exclusive ored to the running keyed message digest

200

204

```
F3_SEED = (((K[(HASH_FOR_KEY(H7,o,var3[6],var4[6],var5[6],var1[6],var0[6],var7[6],var6[6],var2[6],K[(index++%64)]))%64])>>
                (HASH_FOR_KEY(H8,o,var2[7],var6[7],var4[7],var5[7],var3[7],var1[7],var0[7],var7[7],K[(index++%64)]))%25));

F3( F3_SEED )
```

FIG. 12

B

B

204

```
for(rep=0;rep<8;rep++)
{
HASH(H1,var0[rep],var1[rep],var2[rep],var3[rep],var4[rep],var5[rep],var6[rep],var7[rep],K[rep]);
HASH(H1,var1[rep],var2[rep],var3[rep],var4[rep],var5[rep],var6[rep],var7[rep],var0[rep],K[rep+8]);
HASH(H1,var2[rep],var3[rep],var4[rep],var5[rep],var6[rep],var7[rep],var0[rep],var1[rep],K[rep+16]);
HASH(H1,var3[rep],var4[rep],var5[rep],var6[rep],var7[rep],var0[rep],var1[rep],var2[rep],K[rep+24]);
HASH(H1,var4[rep],var5[rep],var6[rep],var7[rep],var0[rep],var1[rep],var2[rep],var3[rep],K[rep+32]);
HASH(H1,var5[rep],var6[rep],var7[rep],var0[rep],var1[rep],var2[rep],var3[rep],var4[rep],K[rep+40]);
HASH(H1,var6[rep],var7[rep],var0[rep],var1[rep],var2[rep],var3[rep],var4[rep],var5[rep],K[rep+48]);
HASH(H1,var7[rep],var0[rep],var1[rep],var2[rep],var3[rep],var4[rep],var5[rep],var6[rep],K[rep+56]);
}
```

205

```
F3_SEED = (((K[(HASH_FOR_KEY(H6,o,var3[6],var4[6],var5[6],var1[6],var0[6],var7[6],var6[6],var2[6],K[(index++%64)]))%64])>>
                    (HASH_FOR_KEY(H5,o,var2[7],var6[7],var4[7],var5[7],var3[7],var1[7],var0[7],var7[7],K[(index++%64)]))%25));
F3( F3_SEED )
```

204

```
for(rep=0;rep<8;rep++)
{
HASH(H2,var0[rep],var2[rep],var3[rep],var4[rep],var5[rep],var6[rep],var7[rep],var1[rep],K[rep]);
HASH(H2,var1[rep],var3[rep],var4[rep],var5[rep],var6[rep],var7[rep],var0[rep],var2[rep],K[rep+8]);
HASH(H2,var2[rep],var4[rep],var5[rep],var6[rep],var7[rep],var0[rep],var1[rep],var3[rep],K[rep+16]);
HASH(H2,var3[rep],var5[rep],var6[rep],var7[rep],var0[rep],var1[rep],var2[rep],var4[rep],K[rep+24]);
HASH(H2,var4[rep],var6[rep],var7[rep],var0[rep],var1[rep],var2[rep],var3[rep],var5[rep],K[rep+32]);
HASH(H2,var5[rep],var7[rep],var0[rep],var1[rep],var2[rep],var3[rep],var4[rep],var6[rep],K[rep+40]);
HASH(H2,var6[rep],var0[rep],var1[rep],var2[rep],var3[rep],var4[rep],var5[rep],var7[rep],K[rep+48]);
HASH(H2,var7[rep],var1[rep],var2[rep],var3[rep],var4[rep],var5[rep],var6[rep],var0[rep],K[rep+56]);
}
```

205

```
F3_SEED = (((K[(HASH_FOR_KEY(H4,o,var3[6],var4[6],var5[6],var1[6],var0[6],var7[6],var6[6],var2[6],K[(index++%64)]))%64])>>
                    (HASH_FOR_KEY(H7,o,var2[7],var6[7],var4[7],var5[7],var3[7],var1[7],var0[7],var7[7],K[(index++%64)]))%25));
F3( F3_SEED )
```

204

```
for(rep=0;rep<8;rep++)
{
HASH(H3,var0[rep],var3[rep],var4[rep],var5[rep],var6[rep],var7[rep],var1[rep],var2[rep],K[rep]);
HASH(H3,var1[rep],var4[rep],var5[rep],var6[rep],var7[rep],var0[rep],var2[rep],var3[rep],K[rep+8]);
HASH(H3,var2[rep],var5[rep],var6[rep],var7[rep],var0[rep],var1[rep],var3[rep],var4[rep],K[rep+16]);
HASH(H3,var3[rep],var6[rep],var7[rep],var0[rep],var1[rep],var2[rep],var4[rep],var5[rep],K[rep+24]);
HASH(H3,var4[rep],var7[rep],var0[rep],var1[rep],var2[rep],var3[rep],var5[rep],var6[rep],K[rep+32]);
HASH(H3,var5[rep],var0[rep],var1[rep],var2[rep],var3[rep],var4[rep],var6[rep],var7[rep],K[rep+40]);
HASH(H3,var6[rep],var1[rep],var2[rep],var3[rep],var4[rep],var5[rep],var7[rep],var0[rep],K[rep+48]);
HASH(H3,var7[rep],var2[rep],var3[rep],var4[rep],var5[rep],var6[rep],var0[rep],var1[rep],K[rep+56]);
}
```

B1

FIG. 12 (CONT'D)

B1

205

```
F3_SEED = (((K[(HASH_FOR_KEY(H2,o,var3[6],var4[6],var5[6],var1[6],var0[6],var7[6],var6[6],var2[6],K[(index++%64)]))%64])>>
                  (HASH_FOR_KEY(H6,o,var2[7],var6[7],var4[7],var5[7],var3[7],var1[7],var0[7],var7[7],K[(index++%64)]))%25));
F3( F3_SEED )
```

204

```
for(rep=0;rep<8;rep++)
{
HASH(H4,var0[rep],var4[rep],var5[rep],var6[rep],var7[rep],var1[rep],var2[rep],var3[rep],K[rep]);
HASH(H4,var1[rep],var5[rep],var6[rep],var7[rep],var0[rep],var2[rep],var3[rep],var4[rep],K[rep+8]);
HASH(H4,var2[rep],var6[rep],var7[rep],var0[rep],var1[rep],var3[rep],var4[rep],var5[rep],K[rep+16]);
HASH(H4,var3[rep],var7[rep],var0[rep],var1[rep],var2[rep],var4[rep],var5[rep],var6[rep],K[rep+24]);
HASH(H4,var4[rep],var0[rep],var1[rep],var2[rep],var3[rep],var5[rep],var6[rep],var7[rep],K[rep+32]);
HASH(H4,var5[rep],var1[rep],var2[rep],var3[rep],var4[rep],var6[rep],var7[rep],var0[rep],K[rep+40]);
HASH(H4,var6[rep],var2[rep],var3[rep],var4[rep],var5[rep],var7[rep],var0[rep],var1[rep],K[rep+48]);
HASH(H4,var7[rep],var3[rep],var4[rep],var5[rep],var6[rep],var0[rep],var1[rep],var2[rep],K[rep+56]);
}
```

205

```
F3_SEED = (((K[(HASH_FOR_KEY(H7,o,var7[5],var5[5],var3[5],var1[5],var6[5],var2[5],var4[5],var0[5],K[(index++%64)]))%64])>>
                  (HASH_FOR_KEY(H1,o,var4[6],var1[6],var6[6],var3[6],var7[6],var0[6],var2[6],var5[6],K[(index++%64)]))%25));
F3( F3_SEED )
```

204

```
for(rep=0;rep<8;rep++)
{
HASH(H5,var0[rep],var5[rep],var6[rep],var7[rep],var1[rep],var2[rep],var3[rep],var4[rep],K[rep]);
HASH(H5,var1[rep],var6[rep],var7[rep],var0[rep],var2[rep],var3[rep],var4[rep],var5[rep],K[rep+8]);
HASH(H5,var2[rep],var7[rep],var0[rep],var1[rep],var3[rep],var4[rep],var5[rep],var6[rep],K[rep+16]);
HASH(H5,var3[rep],var0[rep],var1[rep],var2[rep],var4[rep],var5[rep],var6[rep],var7[rep],K[rep+24]);
HASH(H5,var4[rep],var1[rep],var2[rep],var3[rep],var5[rep],var6[rep],var7[rep],var0[rep],K[rep+32]);
HASH(H5,var5[rep],var2[rep],var3[rep],var4[rep],var6[rep],var7[rep],var0[rep],var1[rep],K[rep+40]);
HASH(H5,var6[rep],var3[rep],var4[rep],var5[rep],var7[rep],var0[rep],var1[rep],var2[rep],K[rep+48]);
HASH(H5,var7[rep],var4[rep],var5[rep],var6[rep],var0[rep],var1[rep],var2[rep],var3[rep],K[rep+56]);
}
```

205

```
F3_SEED = (((K[(HASH_FOR_KEY(H5,o,var7[6],var5[6],var3[6],var1[6],var6[6],var2[6],var4[6],var0[6],K[(index++%64)]))%64])>>
                  (HASH_FOR_KEY(H3,o,var4[7],var1[7],var6[7],var3[7],var7[7],var0[7],var2[7],var5[7],K[(index++%64)]))%25));
F3( F3_SEED )
```

B2

FIG. 12 (CONT'D)

B2

204

```
for(rep=0;rep<8;rep++)
{
HASH(H6,var0[rep],var6[rep],var7[rep],var1[rep],var2[rep],var3[rep],var4[rep],var5[rep],K[rep]);
HASH(H6,var1[rep],var7[rep],var0[rep],var2[rep],var3[rep],var4[rep],var5[rep],var6[rep],K[rep+8]);
HASH(H6,var2[rep],var0[rep],var1[rep],var3[rep],var4[rep],var5[rep],var6[rep],var7[rep],K[rep+16]);
HASH(H6,var3[rep],var1[rep],var2[rep],var4[rep],var5[rep],var6[rep],var7[rep],var0[rep],K[rep+24]);
HASH(H6,var4[rep],var2[rep],var3[rep],var5[rep],var6[rep],var7[rep],var0[rep],var1[rep],K[rep+32]);
HASH(H6,var5[rep],var3[rep],var4[rep],var6[rep],var7[rep],var0[rep],var1[rep],var2[rep],K[rep+40]);
HASH(H6,var6[rep],var4[rep],var5[rep],var7[rep],var0[rep],var1[rep],var2[rep],var3[rep],K[rep+48]);
HASH(H6,var7[rep],var5[rep],var6[rep],var0[rep],var1[rep],var2[rep],var3[rep],var4[rep],K[rep+56]);
}
```

205

```
F3_SEED = (((K[(HASH_FOR_KEY(H6,o,var7[6],var5[6],var3[6],var1[6],var6[6],var2[6],var4[6],var6[6],K[(index++%64)]))%64])>>
                    (HASH_FOR_KEY(H8,o,var4[7],var7[7],var6[7],var3[7],var7[7],var0[7],var2[7],var5[7],K[(index++%64)]))%25));
F3( F3_SEED )
```

```
for(rep=0;rep<8;rep++)
{
HASH(H7,var0[rep],var7[rep],var1[rep],var2[rep],var3[rep],var4[rep],var5[rep],var6[rep],K[rep]);
HASH(H7,var1[rep],var0[rep],var2[rep],var3[rep],var4[rep],var5[rep],var6[rep],var7[rep],K[rep+8]);
HASH(H7,var2[rep],var1[rep],var3[rep],var4[rep],var5[rep],var6[rep],var7[rep],var0[rep],K[rep+16]);
HASH(H7,var3[rep],var2[rep],var4[rep],var5[rep],var6[rep],var7[rep],var0[rep],var1[rep],K[rep+24]);
HASH(H7,var4[rep],var3[rep],var5[rep],var6[rep],var7[rep],var0[rep],var1[rep],var2[rep],K[rep+32]);
HASH(H7,var5[rep],var4[rep],var6[rep],var7[rep],var0[rep],var1[rep],var2[rep],var3[rep],K[rep+40]);
HASH(H7,var6[rep],var5[rep],var7[rep],var0[rep],var1[rep],var2[rep],var3[rep],var4[rep],K[rep+48]);
HASH(H7,var7[rep],var6[rep],var0[rep],var1[rep],var2[rep],var3[rep],var4[rep],var5[rep],K[rep+56]);
}
```

205

```
F3_SEED = (((K[(HASH_FOR_KEY(H3,o,var3[6],var4[6],var5[6],var1[6],var0[6],var7[6],var6[6],var2[6],K[(index++%64)]))%64])>>
                    (HASH_FOR_KEY(H4,o,var2[7],var6[7],var4[7],var5[7],var3[7],var1[7],var0[7],var7[7],K[(index++%64)]))%25));
F3( F3_SEED )
```

204

```
for(rep=0;rep<8;rep++)
{
HASH(H8,var0[rep],var7[rep],var2[rep],var3[rep],var4[rep],var5[rep],var6[rep],var1[rep],K[rep]);
HASH(H8,var1[rep],var0[rep],var3[rep],var4[rep],var5[rep],var6[rep],var7[rep],var2[rep],K[rep+8]);
HASH(H8,var2[rep],var1[rep],var4[rep],var5[rep],var6[rep],var7[rep],var0[rep],var3[rep],K[rep+16]);
HASH(H8,var3[rep],var2[rep],var5[rep],var6[rep],var7[rep],var0[rep],var1[rep],var4[rep],K[rep+24]);
HASH(H8,var4[rep],var3[rep],var6[rep],var7[rep],var0[rep],var1[rep],var2[rep],var5[rep],K[rep+32]);
HASH(H8,var5[rep],var4[rep],var7[rep],var0[rep],var1[rep],var2[rep],var3[rep],var6[rep],K[rep+40]);
HASH(H8,var6[rep],var5[rep],var0[rep],var1[rep],var2[rep],var3[rep],var4[rep],var7[rep],K[rep+48]);
HASH(H8,var7[rep],var6[rep],var1[rep],var2[rep],var3[rep],var4[rep],var5[rep],var0[rep],K[rep+56]);
}
```

B3

FIG. 12 (CONT'D)

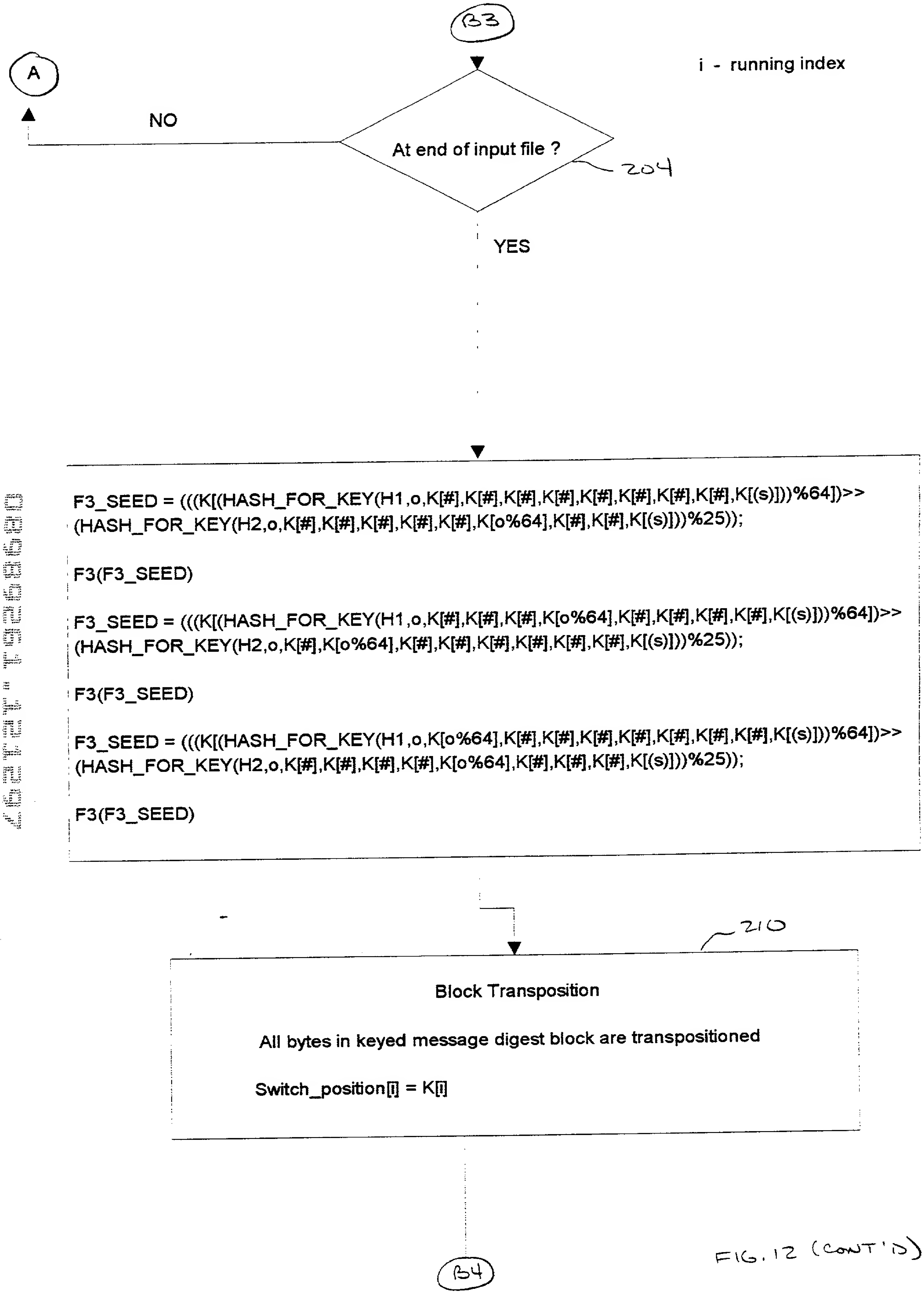

B3
A
i - running index
NO
At end of input file ?
204
YES
F3_SEED = (((K[(HASH_FOR_KEY(H1,o,K[#],K[#],K[#],K[#],K[#],K[#],K[#],K[#],K[(s)]))%64])>>
(HASH_FOR_KEY(H2,o,K[#],K[#],K[#],K[#],K[#],K[o%64],K[#],K[#],K[(s)]))%25));
F3(F3_SEED)
F3_SEED = (((K[(HASH_FOR_KEY(H1,o,K[#],K[#],K[#],K[o%64],K[#],K[#],K[#],K[#],K[(s)]))%64])>>
(HASH_FOR_KEY(H2,o,K[#],K[o%64],K[#],K[#],K[#],K[#],K[#],K[#],K[(s)]))%25));
F3(F3_SEED)
F3_SEED = (((K[(HASH_FOR_KEY(H1,o,K[o%64],K[#],K[#],K[#],K[#],K[#],K[#],K[#],K[(s)]))%64])>>
(HASH_FOR_KEY(H2,o,K[#],K[#],K[#],K[#],K[o%64],K[#],K[#],K[#],K[(s)]))%25));
F3(F3_SEED)
210
Block Transposition
All bytes in keyed message digest block are transpositioned
Switch_position[i] = K[i]
B4

FIG. 12 (CONT'D)

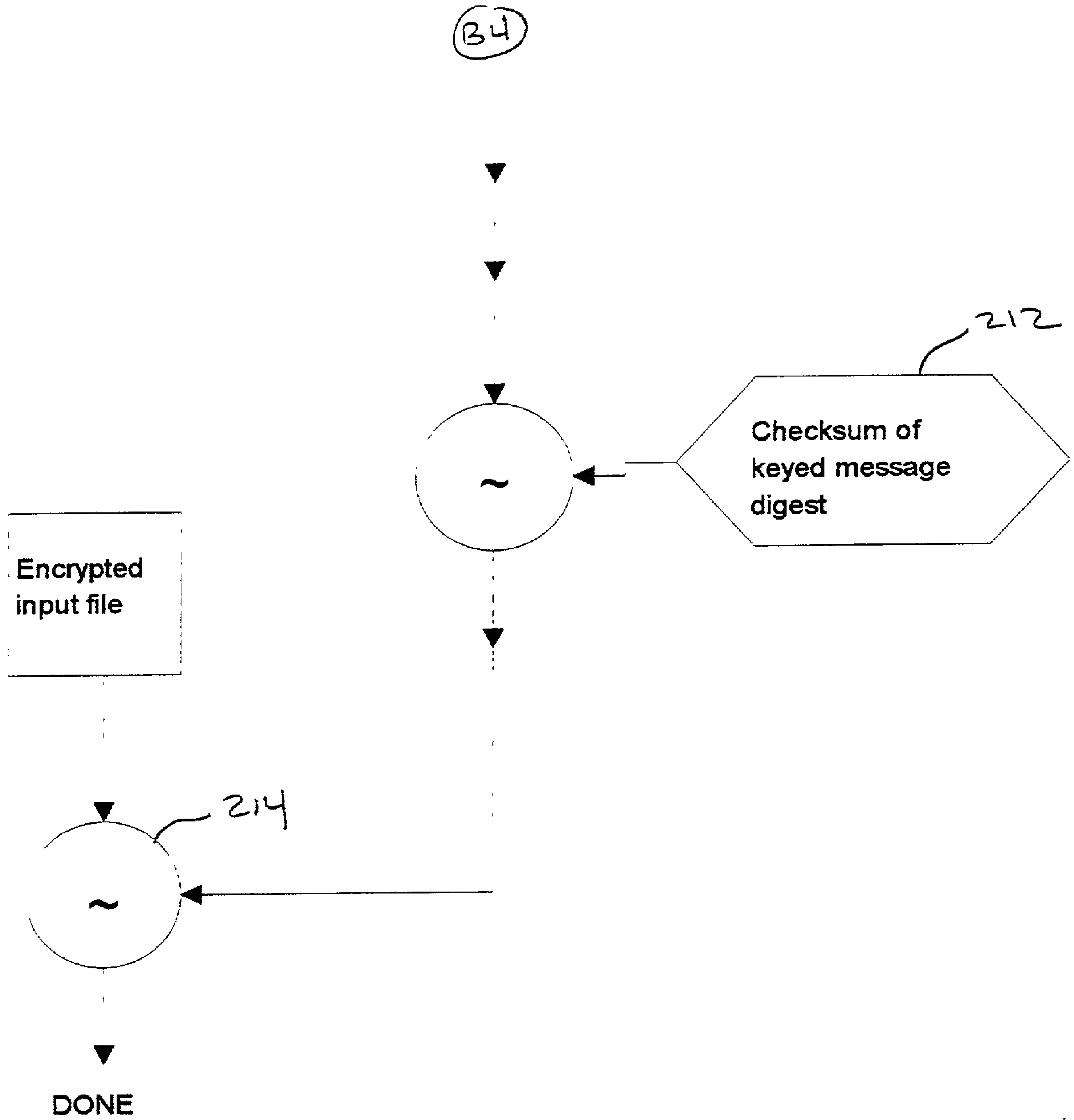

B4
212
Checksum of keyed message digest
~
Encrypted input file
214
~
DONE

FIG. 12 (CONT'D)

PATENT

Attorney's Docket No. 831-2

Applicant or Patentee: Luciano F. Paone

Serial or Patent No.: __________

Filed or Issued: __________

For: COMPUTER IMPLEMENTED SECRET OBJECT KEY BLOCK CIPHER ENCRYPTION AND DIGITAL SIGNATURE DEVICE AND METHOD

VERIFIED STATEMENT (DECLARATION) CLAIMING SMALL ENTITY STATUS (37 CFR 1.9(f) and 1.27(c))--SMALL BUSINESS CONCERN

I hereby declare that I am

[ ] the owner of the small business concern identified below:
[X] an official of the small business concern empowered to act on behalf of the concern

identified below:

NAME OF CONCERN PAONET SOFTWARE, INC.

ADDRESS OF CONCERN 18 Knobhill Drive, Kings Park, New York 11754

I hereby declare that the above identified small business concern qualifies as a small business concern as defined in 13 CFR 121.3-18, and reproduced in 37 CFR 1.9(d), for purposes of paying reduced fees under Section 41(a) and (b) of Title 35, United States Code, in that the number of employees of the concern, including those of its affiliates, does not exceed 500 persons. For purposes of this statement, (1) the number of employees of the business concern is the average over the previous fiscal year of the concern of the persons employed on a full-time, part-time or temporary basis during each of the pay periods of the fiscal year, and (2) concerns are affiliates of each other when either, directly or indirectly, one concern controls or has the power to control the other, or a third-party or parties controls of has the power to control both.

I hereby declare that rights under contract or law have been conveyed, to and remain with the small business concern identified above with regard to the invention, entitled:

COMPUTER IMPLEMENTED SECRET OBJECT KEY BLOCK CIPHER ENCRYPTION AND DIGITAL SIGNATURE DEVICE AND METHOD

by inventor(s) Luciano F. Paone

described in:

[X] the specification filed herewith.
[ ] application serial no. 08/____, filed (Express Mail Label No.)
[ ] patent no. _________, issued ________.

If the rights held by the above identified small business concern are not exclusive, each individual, concern or organization having rights in the invention is listed below* and no rights to the invention are held by any person, other than the inventor, who would not qualify as an independent inventor under 35 CFR 1.9(c) if that person made the invention, or by any concern which would not qualify as a small business concern under 37 CFR 1.9(d) or a nonprofit organization under 37 CFR 1.9(e).

*NOTE: Separate verified statements are required from each named person, concern or organization having rights to the invention averring to their status as small entities. (37 CFR 1.27).

NAME ____

ADDRESS ____
[ ] INDIVIDUAL [ ] SMALL BUSINESS CONCERN [ ] NONPROFIT ORGANIZATION

NAME ____

ADDRESS ____
[ ] INDIVIDUAL [ ] SMALL BUSINESS CONCERN [ ] NONPROFIT ORGANIZATION

I acknowledge the duty to file, in this application or patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small business entity is no longer appropriate. (37 CFR 1.28(b)).

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

NAME OF PERSON SIGNING: ELISA PAONE

TITLE OF PERSON OTHER THAN OWNER: VICE PRESIDENT

ADDRESS OF PERSON SIGNING: 18 KNOBHILL DRIVE, KINGS PARK, NEW YORK 11754

SIGNATURE x Elisa Paone Date x DEC. 12, 1997

Attorney's Docket No. **831-2**

COMBINED DECLARATION AND POWER OF ATTORNEY

(ORIGINAL, DESIGN, NATIONAL STAGE OF PCT, SUPPLEMENTAL, DIVISIONAL, CONTINUATION OR CIP)

As a below named inventor, I hereby declare that:

TYPE OF DECLARATION

This declaration is of the following type: *(check one)*

[X] Original
[ ] Supplemental
[ ] Design

[ ] National Stage PCT
[ ] Divisional
[ ] Continuation
[ ] Continuation-in-Part (CIP)

INVENTORSHIP IDENTIFICATION

NOTE: If the inventors are each not the inventors of all the claims an explanation of the facts, including the ownership of all the claims at the time the last claimed invention was made, should be submitted.

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

COMPUTER IMPLEMENTED SECRET OBJECT KEY BLOCK CIPHER ENCRYPTION AND DIGITAL SIGNATURE DEVICE AND METHOD

the specification of which: *(complete (a), (b) or (c))*

(a) [X] is attached hereto.

(b) [ ] was filed on ________________ as
[ ] Serial No. **08/**_________ or
[ ] Express Mail No. _______________, as Serial No. not yet known
and was amended on ____________. *(If applicable)*

(c) [ ] was described and claimed in PCT International Application No. **PCT/**_____________
filed on _______ and as amended under PCT Article 19 on __________. *(If any)*

ACKNOWLEDGMENT OF REVIEW OF PAPERS AND DUTY OF CANDOR

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above, and that the filing of said specification, if heretofore filed, was authorized by me.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, §1.56(a).

CLAIM OF PRIORITY OF EARLIER FOREIGN APPLICATION(S) UNDER 35 U.S.C. §119(a)-(d)

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate or of any PCT international application(s) designating at least one country other than the United States of America listed below and have also identified below any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed:

(List prior foreign/PCT application(s) filed within 12 months (6 months for design) prior to this U.S. application.)

NOTE: *Where item (c) is entered above and the International Application which designated the U.S. claimed priority check item (e), enter the details below and make the priority claim.*

COUNTRY (or PCT)	APPLICATION NO.	DATE OF FILING *(Day/Month/Year)*	PRIORITY CLAIMED UNDER 35 USC §119
			[ ]YES [ ]NO
			[ ]YES [ ]NO

CLAIM FOR BENEFIT OF PRIOR U.S. PROVISIONAL APPLICATION(S) UNDER 35 U.S.C. §119(e)

I hereby claim the benefit under Title 35, United States Code, §119(e) of any United States provisional application(s) listed below:

(List prior U.S. provisional applications.)

PROVISIONAL APPLICATION NO.	FILING DATE *(Day/Month/Year)*

CLAIM FOR BENEFIT OF EARLIER U.S./PCT APPLICATION(S) UNDER 35 U.S.C. 120

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) or PCT international application(s) designating the United States of America that is/are listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in such prior application(s) in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §1.56(a) which occurred between the filing date of the prior application(s) and the national or PCT international filing date of this application:

(List prior U.S. applications or PCT international applications designating the U.S. for benefit under 35 U.S.C. §120.)

U.S. APPLICATIONS **STATUS** *(Check One)*

U.S. SERIAL NO.	U.S. FILING DATE *(Day/Month/Year)*	Patented	Pending	Abandoned
0 /		[ ]	[ ]	[ ]
0 /		[ ]	[ ]	[ ]

PCT APPLICATIONS DESIGNATING THE U.S. **STATUS** *(Check One)*

PCT APPLN. NO.	PCT FILING DATE *(Day/Month/Year)*	U.S. SERIAL NOS. ASSIGNED (If any)	Patented	Pending	Abandoned
PCT/			[ ]	[ ]	[ ]
PCT/			[ ]	[ ]	[ ]

35 USC 119 PRIORITY CLAIM, IF ANY, FOR ABOVE LISTED U.S./PCT APPLICATIONS

PRIORITY APPLICATION NO.	PRIORITY COUNTRY	FILING DATE *(Day/Month/Year)*	ISSUE DATE *(Day/Month/Year)*

POWER OF ATTORNEY

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office in connection therewith:

Charles R. Hoffmann, Reg. No. 24,102; Ronald J. Baron, Reg. No. 29,281; Gerald T. Bodner, Reg. No. 30,449; Alan M. Sack, Reg. No. 31,874; A. Thomas Kammer, Reg. No. 28,226; Arlene D. Morris, Reg. No. 32,657; R. Glenn Schroeder, Reg. No. 34,720; Glenn T. Henneberger, Reg. No. 36,074; Livia Boyadjian, Reg. No. 34,781; Sean W. O'Dea, Reg. No. 37,690; Lindsay S. Adams, Reg. No. 36,425; Paul J. Otterstedt, Reg. No. 37,411; Irving N. Feit, Reg. No. 28,601; Paul D. Ackerman, Reg. No. 39,891; Jessica H. Tran, Reg. No. 40,846; Anthony E. Bennett, Reg. No. 40,910, each of them of HOFFMANN & BARON, LLP, 350 Jericho Turnpike, Jericho, New York 11753; and Daniel A. Scola, Jr., Reg. No. 29,855; Salvatore J. Abbruzzese, Reg. No. 30,152; Kirk M. Miles, Reg. No. 37,891; Kevin C. Hooper, Reg. No. 40,402; and Robert F. Chisholm, Reg. No. 39,939, each of them of HOFFMANN & BARON, LLP, 1055 Parsippany Boulevard, Parsippany, New Jersey 07054.

PLEASE SEND CORRESPONDENCE TO:	PLEASE DIRECT TELEPHONE CALLS TO:
Glenn T. Henneberger, Esq. HOFFMANN & BARON, LLP 350 Jericho Turnpike Jericho, NY 11753	Glenn T. Henneberger, Esq. (516) 822-3550

DECLARATION

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

SIGNATURE(S)

Full Name of Sole or First Inventor:	Luciano F. Paone
Country of Citizenship:	U.S.A.
Residence Address:	18 Knobhill Drive, Kings Park, New York 11754
Post Office Address:	**Same as Above**

Date: x Dec 12, 1997 Inventor's signature x Luciano F. Paone

Full Name of Second Joint Inventor:	
Country of Citizenship:	U.S.A.
Residence Address:	
Post Office Address:	**Same as Above**

Date: ____________ Inventor's signature ____________

NOTE: All above spaces identifying inventors must be completed or deleted before any inventor executes this application